Logo:
Programming
with
Turtle Graphics

International Business Machines Corporation   Boca Raton, Florida 33432

IBM Program License Agreement

YOU SHOULD CAREFULLY READ THE FOLLOWING TERMS AND CONDITIONS BEFORE OPENING THIS DISKETTE(S) OR CASSETTE(S) PACKAGE. OPENING THIS DISKETTE(S) OR CASSETTE(S) PACKAGE INDICATES YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS. IF YOU DO NOT AGREE WITH THEM, YOU SHOULD PROMPTLY RETURN THE PACKAGE UNOPENED; AND YOUR MONEY WILL BE REFUNDED.

IBM provides this program and licenses its use in the United States and Puerto Rico. You assume responsibility for the selection of the program to achieve your intended results, and for the installation, use and results obtained from the program.

LICENSE

You may:

a. use the program on a single machine;

b. copy the program into any machine readable or printed form for backup or modification purposes in support of your use of the program on the single machine (Certain programs, however, may include mechanisms to limit or inhibit copying. They are marked "copy protected.");

c. modify the program and/or merge it into another program for your use on the single machine (Any portion of this program merged into another program will continue to be subject to the terms and conditions of this Agreement.); and,

d. transfer the program and license to another party if the other party agrees to accept the terms and conditions of this Agreement. If you transfer the program, you must at the same time either transfer all copies whether in printed or machine-readable form to the same party or destroy any copies not transferred; this includes all modifications and portions of the program contained or merged into other programs.

You must reproduce and include the copyright notice on any copy, modification or portion merged into another program.

YOU MAY NOT USE, COPY, MODIFY, OR TRANSFER THE PROGRAM, OR ANY COPY, MODIFICATION OR MERGED PORTION, IN WHOLE OR IN PART, EXCEPT AS EXPRESSLY PROVIDED FOR IN THIS LICENSE.

IF YOU TRANSFER POSSESSION OF ANY COPY, MODIFICATION OR MERGED PORTION OF THE PROGRAM TO ANOTHER PARTY, YOUR LICENSE IS AUTOMATICALLY TERMINATED.

TERM

The license is effective until terminated. You may terminate it at any other time by destroying the program together with all copies, modifications and merged portions in any form. It will also terminate upon conditions set forth elsewhere in this Agreement or if you fail to comply with any term or condition of this Agreement. You agree upon such termination to destroy the program together with all copies, modifications and merged portions in any form.

LIMITED WARRANTY

THE PROGRAM IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU (AND NOT IBM OR AN AUTHORIZED PERSONAL COMPUTER DEALER) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
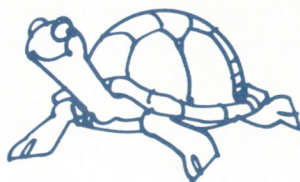
Logo:
Programming with Turtle Graphics

Preface

Logo: Programming with Turtle Graphics
introduces Logo to new IBM Personal
Computer users and to people who know about
computers. It does not explain all possible uses
of Logo. It shows how to build and change
programs and how to store and retrieve them
when needed. Even if you are familiar with
other programming languages or do not have the
special equipment needed to try the turtle
graphics commands, you should read through
this book to get an idea of how Logo programs
work.

We have included many pictures in the margin
to illustrate the examples in the text. You'll
notice some green text in the book. We use this
color to show what the computer types and what
you type on the screen.

Most chapters have a section at the end entitled
"Logo Vocabulary." It contains a list of
primitives introduced in the chapter. Most
chapters also have a section called "Special
Keys." It lists the keys introduced in the
chapter.

We discuss some of the error messages that
Logo prints when you do something Logo
doesn't understand. For a complete list of error
messages and more details on Logo
programming, see the *IBM Personal Computer
Logo Reference.*

Note that the terms "disk" and "diskette" are used in this book. Where "diskette" is used, it applies only to diskette drives and diskettes. Where "disk" is used, it applies to both the IBM nonremovable fixed disk drives and diskettes.

Contents

vii

Chapter 1. Introduction

Contents

Logo is a language for computers. Compared with spoken languages like English or French, Logo has a very small number of words and rules of grammar. Using Logo with your IBM Personal Computer, you can do things you couldn't dream of doing with ordinary languages–things like inventing computer games and drawing circles, flowers, stars, or almost any figure you can imagine. In fact, you'll be surprised by what Logo and you can do together, once you know how to make it work for you.

Programming in Logo can be a great game in which you create things on a computer screen and then use them to make other new things. While doing this, you'll have fun and solve problems.

Maybe you know about computer graphics (drawings) through your exposure to video games and special effects in movies. We're convinced that the best way to find out about programming with Logo is to first learn how to make computer graphics. That way you can *see* how your programs work.

The Logo Turtle

In Logo, the graphics are drawn by a turtle that is shaped like a triangle. The Logo turtle moves around on the computer screen with a pen. It does whatever you tell it to do. Of course, the Logo turtle only understands the Logo language, so to get it to draw your graphics you have to learn Logo. That's what this book is all about.

Using Logo with Your IBM Personal Computer

In order to use Logo, you need:

- An IBM Personal Computer XT or an IBM Personal Computer with a minimum of 128K bytes of random access memory.

- One diskette drive (minimum).

- A blank diskette.

- One or both of the following:

  — An IBM Color/Graphics Monitor Adapter with any color or black-and-white monitor or TV to do turtle graphics. An IBM Color Display or any direct-drive monitor capable of displaying 40 or more columns is recommended to achieve quality turtle graphics.

    Note: Composite color monitors or televisions may not produce true color images.

— An IBM Monochrome Display and
Parallel Printer Adapter and
monochrome monitor.

Note: You cannot use an IBM
Monochrome Display for turtle
graphics.

• The Logo Language Diskette, which is
found at the back of the *Logo Reference*
binder. A Logo Language Diskette
contains the Logo program with samples
and useful tools, the IBM Personal
Computer Disk Operating System (DOS)
Version 2.00, the FORMAT,
DISKCOPY, and GRAPHICS
commands, and AUTOEXEC.BAT, a
batch file which is automatically executed
when you start the system.

• A desire to learn, to try things, to work
hard, and enjoy yourself.

A printer may be useful at times, but it's
not required.
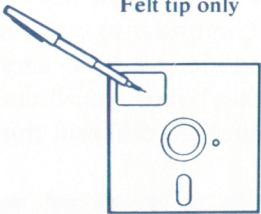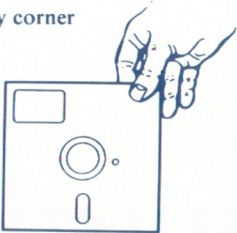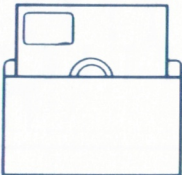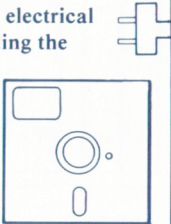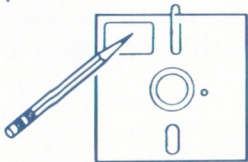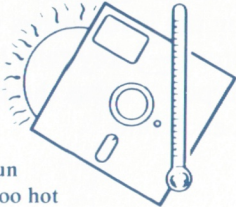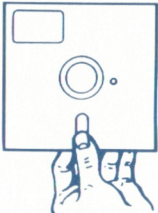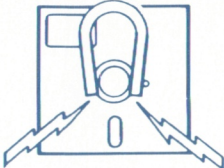
Caring for your Diskettes

Diskettes are coated with magnetic material that memorizes programs. Parts of programs can be lost if this magnetic surface is damaged by heat, cold, or handling. To keep your diskettes in good shape you should:

● Always put the diskettes back into their protective jackets after use.

● Never touch diskettes through the window slots, where the magnetic surface of the diskette is exposed.

● Keep diskettes out of direct sunlight and away from other sources of heat. Also, keep diskettes out of very cold areas. Diskettes may be stored at temperatures ranging from 40 to 125 degrees Fahrenheit (4 to 52 degrees Celsius).

● Use a felt-tip pen when writing on diskette labels to avoid damaging the magnetic surface of the diskette inside the envelope.

Important Tips

Do's

Felt tip only	**Grasp by corner**
When not in use	**Keep it away from electrical equipment (including the telephone and the TV).**

Don'ts

No pencils No clips No ballpoints	Don't bend
No sun Not too hot	Don't touch diskette
No magnets	No cleaning

Your IBM Personal Computer Keyboard

Because you have to use the keyboard of your IBM Personal Computer to communicate with Logo, it's a good idea to get to know it well. Looking at the keyboard, you'll notice it has various keys which do different things.

Function keys   **Typewriter keyboard**   **Numeric keypad**
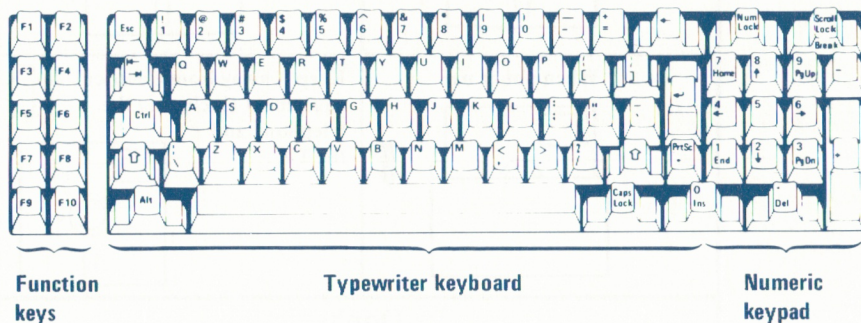
Character Keys
The character keys are letters of the alphabet, numbers, and punctuation marks.

Enter (←) Key
You use the Enter (←) key every time you want Logo to obey the instructions you've given it. In fact, the Enter (←) key tells Logo: "Now do what I just typed." Press the Enter (←) key after typing each line.

Shift ⬆ Key

There is a Shift ⬆ key at each end of the keyboard. When you press the Shift ⬆ key alone, nothing happens. But holding down the Shift ⬆ key while pressing certain character keys changes these character keys. For example, if you hold down the Shift ⬆ key and then press the ⟦⟧ key at the top of the keyboard, a dollar sign ($) is produced on the screen.

For the Shift ⬆ key to have an effect, always press the Shift ⬆ key first and then hold it down while typing the character key.

> **Note:** When typing the alphabet characters (A, B, C, etc.), you must always type on the keys *without* shifting them. Shifting them will make the letters appear on the screen in lowercase, and Logo understands only capital letters.

Control Ctrl Key

As with the Shift ⬆ key, when you press and hold the Ctrl key and press certain character keys, you change what they do. For example, pressing the Ctrl key with the C key produces a Heart ♥ on the screen. But, unlike the Shift ⬆ key, a combination of the Ctrl key and certain character keys has an important effect that you cannot see on the screen.

For example, while holding down the ⎡Ctrl⎤ key, press the key labeled Break (also labeled ⎡Scroll Lock⎤). The combination of the Ctrl-Break key, called *Break*, tells Logo to stop what it's doing. Logo lets you know it has done so by printing

STOPPED!
?■

on the screen.

Another important use of the ⎡Ctrl⎤ key is in the ⎡Ctrl⎤ - ⎡Alt⎤ - ⎡Del⎤ key combination. This three-key combination performs a *system reset*, which is similar to switching the computer from off to on. If your Logo Language Diskette is in drive A at the time, the Logo program is loaded into memory.

We'll introduce other uses of the ⎡Ctrl⎤ key later.

Escape ⎡Esc⎤ Key
The ⎡Esc⎤ key is used in Logo to exit from the Logo editor. This key is discussed along with other special editing keys in Chapter 3.

Bracket Keys
The left bracket ⎡[⎤ and right bracket ⎡]⎤ keys are used in Logo to group different kinds of information together. You'll learn more about the use of these keys later.

Parentheses Keys

The left parenthesis ⎡₉⎤ is produced by pressing the Shift ⎡⇧⎤ key and the 9 key at the top of the keyboard. The right parenthesis ⎡₀⎤ is produced by pressing the Shift ⎡⇧⎤ key with the zero 0 key. Remember the difference between the ( ) (parentheses) and [ ] (brackets) in Logo; you will learn why in this book.

Cursor Keys

The cursor is the flashing box you will see often on the screen. On the right-hand side of the keyboard are keys with numbers. In Logo, four of these keys act as cursor keys. Cursor keys use shortcuts to place the cursor where you want it quickly.

The four cursor keys on the number keypad will be a great help to you in writing and correcting anything you type. You should get to know them well.

Notice the arrow pointing up on the 8 key. We call this key the Cursor Up ⎡↑⎤ key; it moves the cursor up to the previous line. You can only use this key in the Logo editor.

On the 2 key, there is an arrow pointing down. This is called the Cursor Down ⎡↓⎤ key. It moves the cursor down to the next line. You can only use this key in the Logo editor.

On the 6 key, there's an arrow pointing right, the Cursor Right ⟨6→⟩ key. It moves the cursor one space to the right (forward).

And on the 4 key, there's an arrow pointing left, the Cursor Left ⟨4←⟩ key. It moves the cursor one space to the left (backward).

Num Lock ⟨Num Lock⟩ Key

You can use the ⟨Num Lock⟩ key to set the number keypad so it works more like a calculator keypad. Pressing the ⟨Num Lock⟩ key shifts the number keypad into its own uppercase mode, so that you get the numbers 0 through 9 and the decimal point, as indicated on the keytops. Pressing ⟨Num Lock⟩ again will return the keypad to its normal cursor control mode.

Spacebar

The Spacebar prints an invisible (but very important) character called a space. Logo uses spaces to separate words. For example, Logo reads THISISAWORD as a single word and reads THIS IS A WORD as four words.

Note that pressing the Spacebar inserts a space at the cursor's position. Thus, pressing the Spacebar when the cursor is on the O of D◘G produces D ◘G.

Backspace ⬅ Key

You must not confuse the Cursor Left ⬅ cursor key with the Backspace ⬅ key, the large key on the top row of the keyboard. When you press this key, it goes backwards along a line and erases all characters it passes over.

Delete Del Key

Pressing the Del key once erases the character under the cursor. If you hold the Del key down, it will continue deleting (erasing) characters until you release the key.

Function Keys

The keys labeled F1 to F10, located on the far left of the keyboard, are called *function keys*. Only the F1, F2, F3, F4, and F5 keys have special meaning in Logo. You'll learn more about these keys later.

The "Special Keys" section, at the end of some chapters, contains a list of special keys introduced within the chapter.

The Importance of Accurate Typing

To get Logo to do things for you, you have to spell words correctly and put in the proper spacing where necessary. If you don't, Logo will usually respond with

I DON'T KNOW HOW TO *name*

where *name* is whatever Logo is having trouble with.

When that happens, retype your instructions correctly. For details about making corrections, see "Correcting Typing Mistakes" in Chapter 2.

Logo Primitives

To create things on the screen, you must tell the turtle what you want it to do in Logo's language. Logo language is made up of about 200 words called *primitives*. Primitives cause the computer to perform certain basic actions such as putting words and lists of words together, adding and subtracting numbers, and drawing lines on the screen.

Logo primitives are made up of operations (like SUM) and commands (like FORWARD). Throughout this book, you'll be concerned mainly with commands.

Using primitives like building blocks, you can make new primitives and then use these to make more new primitives.

Many of the primitives have short forms, which can be used instead of the longer forms. When primitives are first introduced in this book, their short forms are shown in parentheses, for example FORWARD (or FD).

Some of the most important Logo primitives are:

Primitive	Short Form
TO	
FORWARD	FD
BACK	BK
PRINT	PR
EDIT	ED
RIGHT	RT
LEFT	LT

You'll get to know more of these Logo primitives as we go along. The "Logo Vocabulary" section, at the end of most chapters, contains a list of primitives and their short forms introduced in the chapter.

Bug Boxes

From time to time in this book, you'll see a bug in the margin, with the words "Bug Box" beside it. A *bug* is a name for things that don't work the way you expect them to. The Bug Box tries to warn you of what may go wrong and why this may happen. It suggests ways to avoid or fix problems. We have not used green in the bug boxes to indicate what the computer types and what you type.

Little bugs also appear in the margin to tell you that a Bug Box is coming up.

Starting Logo

1. Lift the tab to open the door of drive A (the left-hand drive) on your IBM Personal Computer.

Drive A

Logo Diskette   System
                     Unit
                     Switch

Drive A   Drive B

Logo Diskette   System
                     Unit
                     Switch
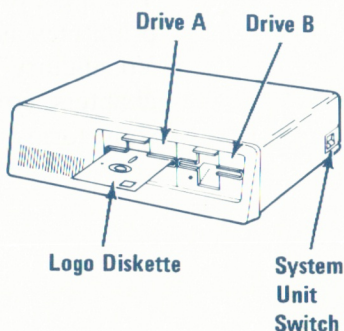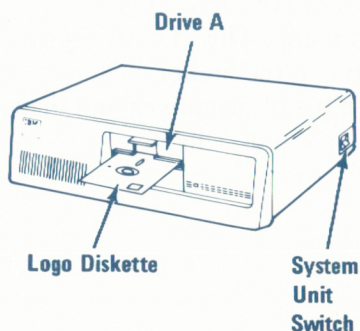
2. Holding your Logo Language Diskette by the end with the label, insert it into drive A with the label facing up; then pull the tab down to close the drive door.

3. Find the System Unit Switch on your IBM Personal Computer. (It's on the right-hand side of the computer, at the back.) Switch it to On. Switch on your monitor or TV.

   If your computer is already On, hold down the Ctrl and Alt keys and at the same time press the Del key. As already noted, this three-key combination performs a system reset.

4. The drive A light will go on. This lets you know the drive is loading DOS into the computer's memory. This takes a few seconds. Then the drive light will go off and you will see a message asking you to enter today's date.

   A>DATE
   Current date is Tue 1-01-1980
   Enter new date:

   If today were June 20, 1983, you would type 6-20-1983 and press the Enter ⏎ key. Enter the date now.

The drive light will go on and you will then
see a message asking you to enter the time.

A>TIME
Current time is 0:00:10.31
Enter new time:

You use the 24-hour clock to record time.
If, for example, the time were 10:30 a.m.,
you would type 10:30 and press the
Enter ⏎ key. If it were 4:30 p.m., you
would type 16:30. Enter the time now.

The drive light will go on, and you will see

A>GRAPHICS

A>LOGO

5.   After a few seconds, the drive light will go
      off and the copyright statement and serial
      number will appear on the screen as well
      as a message saying

      WELCOME TO LOGO
      ? ▮

Below the message you will see a ? (question
mark), the prompt symbol, followed by a
flashing box ■.

When ? (question mark) is on the screen, Logo
waits for you to type something. Whenever you
type in response to the prompt symbol, you are
at *top level*. Top level means Logo is not
currently executing a procedure and is ready to
accept your commands.

The flashing box ■ is the *cursor*. It appears
when Logo wants you to type something and
shows where the next character you type will
appear.

Bug Box
Possible Bugs

1. If you have problems starting up Logo, try
    the following:

    ●   If you have removed the Logo
        Language Diskette, reinsert it into
        drive A.

    ●   Make sure the diskette is properly
        inserted. The label should face up, at
        the front of the drive (near your
        hand).

    ●   Switch off the computer, wait five
        seconds, and switch it on again.

    ●   If you still have problems, see the
        *IBM Personal Computer Guide to
        Operations*.

2. If you can't see the first few letters of
    "WELCOME TO LOGO" for example,

    LCOME TO LOGO

    type

    SETWIDTH 40 2

The second number (2) indicates the number of letters missing from the beginning of the word "WELCOME". This will move the text line over to the right by the specified number of spaces. (For further information, see the SETWIDTH command in the *Logo Reference.*)

Before You Begin Using Logo

If your IBM Logo Language Diskette is being used for the first time, it is important that you make a backup copy (duplicate) and store your master in a safe place. Always use your backup diskette. If you happen to lose or damage this diskette, you can retrieve your master and make a new backup.

In the next chapter, you're going to start using Logo. Later on, you will want to save the things you're about to create on the screen. To be able to do that, you will need a formatted file diskette.

Backup your Language Diskette and format your file diskette now. See Appendix A for instructions.

Chapter 2. Introducing the Turtle

Contents

Logo has many commands to control the turtle. In this chapter, you'll get to know some of the most important turtle commands.

SHOWTURTLE (or ST) Command

The turtle is hiding on the screen. To make it appear, you use the command SHOWTURTLE (or ST).
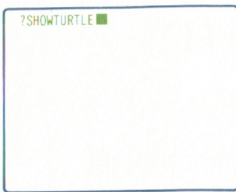
`?SHOWTURTLE ■`

Without putting a space between SHOW and TURTLE, type
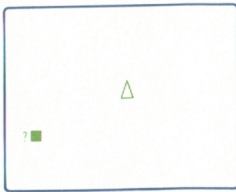
SHOWTURTLE

Now press the Enter ⬅ key, and the turtle will appear as a triangle pointing straight up in the center of the screen. This is called the turtle's *startup state*. You can get the same result by using ST, the short form for SHOWTURTLE.

△

?■

SHOWTURTLE

Every time the turtle appears on the screen, it has a specific *position* and a specific *heading*:

● The *position* is where the turtle is placed on the screen.

● The *heading* is the direction in which the turtle is pointing.

● The position and heading together are called the turtle's *state*.

The most important turtle commands are those that change the turtle's state.

After you type the command SHOWTURTLE, the ? (prompt symbol) and the Cursor ■ go from the top of the screen to the lower part of the screen (actually to the 6th line from the bottom of the screen). From now on, your typed instructions will appear only on the last six lines of the screen, leaving the rest of the screen free for graphics.

HIDETURTLE (or HT) Command

You can make the turtle invisible by typing HIDETURTLE (or HT). Type

HIDETURTLE (or HT)

Press the Enter ⏎ key.

To get the turtle back, type

SHOWTURTLE (or ST)

Press the Enter ⏎ key.

Bug Box
Possible Bugs

1. If you don't have an IBM Color/Graphics Monitor Adapter in your computer, Logo can't carry out any of these turtle graphics commands. In this case, we still suggest you read through this book before going on to the *Logo Reference*.

2. You can work with an IBM Monochrome Display and a monitor or TV at the same time. In this case, you must give the following command to use turtle graphics:

   .SETSCREEN 2

   A turtle will appear on your monitor or TV. All graphics will be executed on the monitor or TV, and all text will continue to appear on your IBM Monochrome Display.

3. If you make a spelling mistake or put a space between SHOW and TURTLE and press the Enter ⏎ key, Logo responds

   I DON'T KNOW HOW TO TURTLE

Type

SHOWTURTLE (or ST)

again correctly.

If you notice the error before you press the
Enter [←] key, you can make the
correction right away. For more
information about this kind of bug, you can
read the section called "Correcting Typing
Mistakes" in this chapter.

Changing the Turtle's State (Position and Heading)

To change the turtle's position and heading, we
use the turtle commands FORWARD, RIGHT,
BACK, and LEFT.

FORWARD (or FD) Command

The command FORWARD (or FD) tells the
turtle to move forward. After typing
FORWARD, you must also type a number
telling how many steps forward it must move.
This number is called an *input*. If your input
numbers are very large, the turtle may go off the
screen and then reappear on the opposite side of
the screen. This can produce some surprising
results.
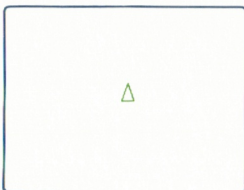
You must *always* type a space between a
command and an input. If you type more than
one space between a command and an input,
that's okay. Logo will ignore the extra spaces.

Δ

Although you can use any input number you want, let's use an input of 50. Type
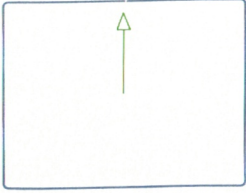
FORWARD 50 (or FD 50)

Press the Enter (←) key.

You'll notice that the turtle changes its position but not its heading.

RIGHT (or RT) Command

To change the turtle's heading to the right, we tell it to turn **RIGHT** (or **RT**) a number of degrees. In the following example, we tell the turtle to turn 90 degrees to the right. Type
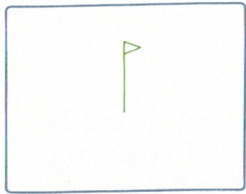
RIGHT 90 (or RT 90)

Press the Enter (←) key.

Notice that the turtle changes its heading but not its position on the screen.

BACK (or BK) Command

BACK (or **BK**) tells the turtle to back up from its current position. Let's use an input of 50 again. Type
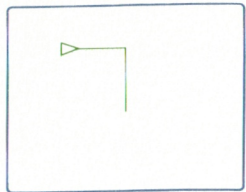
BACK 50 (or BK 50)

Press the Enter (←) key.

The turtle changes it position but not its heading.

LEFT (or LT) Command

🕷 To change the turtle's heading to the left, we tell it to turn LEFT (or LT) a number of degrees. Type

`LEFT 45` (or `LT 45`)

Press the Enter ⏎ key.

The turtle turns 45 degrees to the left of where it had been heading. The turtle changes only its heading, not its position.

You can see the effect of the turn more clearly if you tell the turtle to go FORWARD 25. Type

`FORWARD 25` (or `FD 25`)

Press the Enter ⏎ key.

Try playing with the commands that change the turtle's state. For example, with RT or LT, try different inputs greater or less than 90 and see how much the turtle will turn. How many degrees will make the turtle turn around and face the opposite direction? How many degrees will make the turtle turn all the way around and face the same direction again? If you don't know, try different numbers to find out.

CLEARSCREEN (or CS) Command

The screen is now full of lines where the turtle has been. We call these lines *turtle tracks*. To clear the screen of turtle tracks, use the command CLEARSCREEN (or CS). When the screen clears, the turtle will appear back in the center of the screen and will be heading straight up. Type

CLEARSCREEN (or CS)

Press the Enter ⏎ key.

Notice that the text is not cleared.

```
                △
?RT 90
?BK 50
?LT 45
?FD 25
?CLEARSCREEN
? ■
```
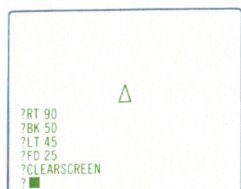
Bug Box
While you're trying out the commands, the turtle may go off the edge of the screen and then come back in an unexpected place. That's because you've used a very large input number for FORWARD or BACK. Type CLEARSCREEN to move the turtle back to the center of the screen.

Correcting Typing Mistakes

When using Logo, you may make typing mistakes. The most common typing mistake is not spacing between a command and its input. For example, if you type FORWARD50, Logo will *not* understand it as FORWARD with an input of 50 (which, of course, is FORWARD 50) and will reply

I DON'T KNOW HOW TO FORWARD50

There are two things you can do to fix a typing mistake.

If you spot an error *after* pressing the Enter ⟵ key, retype the command correctly.

If you see the error *before* pressing the Enter ⟵ key, correct it by pressing the Cursor Left ⟵ key as many times as needed to move the Cursor ■ back along the line to where the error is. (If you hold the Cursor Left ⟵ key down, it will continue to move.)

When you correct a spelling mistake by typing in the correct character, the error will remain on the screen. You erase it by using the ⌨Del key. Be careful not to hold down the ⌨Del key or you will erase other characters. If a space is missing, you can insert it by pressing the spacebar.

To change FORWARD50 to FORWARD 50, here's what you do:

FORWARD50 ▐

Press the Cursor Left ⌨⁴← key twice, which puts the cursor over the 5.

FORWARD⬛0

Press the spacebar once. The command will now appear as
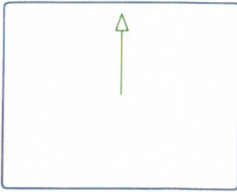
FORWARD ⬛0

Press the Enter ⌨← key.

The turtle has responded to the command by moving forward 50 steps on the screen.

You could also misspell words. For example, you could type FRWARD instead of FORWARD. The difference between FRWARD and FORWARD is only an O, but to Logo it makes a big difference. If you type FRWARD 50, Logo won't know what to do and will reply

I DON'T KNOW HOW TO FRWARD

If you haven't pressed the Enter ⏎ key yet, while this is on the screen,

FRWARD 50 ■

press the Cursor Left ⏎ key eight times, which puts the cursor over the R.

F**R**WARD 50

Type O. The command will now appear as

FO**R**WARD 50

Press the Enter ⏎ key. The turtle responds by moving forward 50 steps on the screen.

Drawing a Square

Using FORWARD and RIGHT (or LEFT), make the turtle draw a square. For convenience, use the short forms FD, RT, and LT. Type the following and remember to press Enter ⏎ after each command.

CS
FD 30
RT 90
FD 30
RT 90
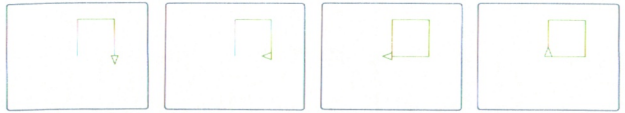
```
FD 30
RT 90
FD 30
RT 90
```

If you used FD 50 instead of FD 30, the turtle would draw a bigger square. You could use larger numbers, but remember that if you use a very large number the turtle may go off the screen.

To get the turtle to draw a square in the opposite direction, use LT instead of RT.

Pen Commands

You can change the state of the turtle's pen with the commands PENUP, PENDOWN, PENERASE, and PENREVERSE.

PENUP (or PU) Command

PENUP (or PU) tells the turtle to move without drawing lines.

PENDOWN (or PD) Command

PENDOWN (or PD) tells the turtle to draw lines again. Type the following and remember to press Enter ⏎ key after each command.
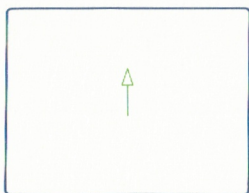
CS
FD 30
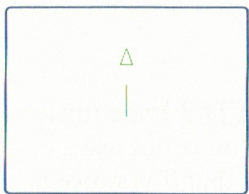
PU
FD 20
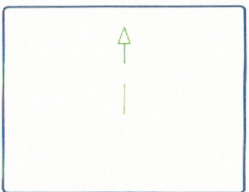
PD
FD 20

PENERASE (or PE) Command

Type the command PENERASE (or PE), and
the turtle's pen becomes an eraser instead of a
drawing tool. Now make the turtle go over a line
it has drawn, and the line will be erased. Clear
the screen (by typing CS), make sure the pen is
down (PD), and then type the following
commands:
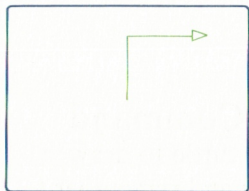
FD 50
RT 90
FD 50
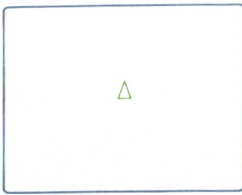
Now type

```
PE
BK 50
LT 90
BK 50
```

The turtle will erase any lines it goes over until told to **PENUP (PU)** or to **PENDOWN (PD)**. Notice that with **PENERASE** the turtle doesn't draw any new lines.

PENREVERSE (or PX)
Command

PENREVERSE (or PX) is a mixture of PENDOWN and PENERASE. This command actually does an exchange: the turtle will draw a line whenever it moves over a blank background. But, if the turtle moves over a drawn line, it will erase it. This can be used to produce some surprising effects. Watch what happens when you type each line as follows:
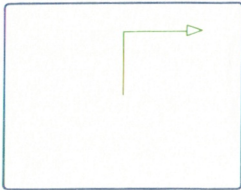
```
PX
FD 50
RT 90
FD 50
```
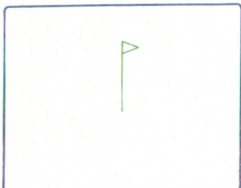
Make the turtle go back. Type

```
BK 50
```

It erased the line!
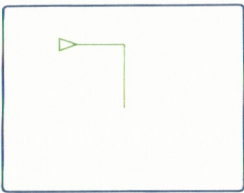
Let's see what happens if the turtle backs up again. Type

`BK 50`

It drew a line! To return the pen to its normal state, type

`PD`

Logo Vocabulary

Here is a list of commands used in this chapter, along with their short forms.

Command	Short Form
BACK	BK
CLEARSCREEN	CS
FORWARD	FD
HIDETURTLE	HT
LEFT	LT
PENDOWN	PD
PENERASE	PE
PENREVERSE	PX
PENUP	PU
RIGHT	RT
SHOWTURTLE	ST

Special Keys

Here is a list of special keys introduced in this chapter.

Cursor Left ⬚

Delete ⬚

Enter ⬚

Spacebar

Chapter 3. Defining Procedures

Contents

Programming in Logo is based on the idea of building blocks. Put building blocks together, and you can form a single block with its own name. When you want to use this block, call it by its name instead of building it again. This block also can be used as part of another block with another name.

By using Logo primitives, you can make up or "define" a series of instructions to produce the result you want. This series of instructions is called a *procedure*. You can give this procedure any name you want.

There are two ways to define a procedure in Logo. One uses the command TO and the other uses the command EDIT (or ED).

Defining a Procedure with TO

Let's draw a letter "T". Type
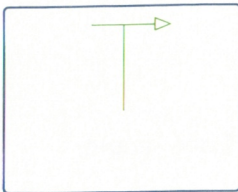
```
FD 50
RT 90
BK 20
FD 40
```

If we want to draw the "T" in the future, it is a good idea to define a procedure to do this. Let's call the procedure TEE. After we type the instructions once, we can simply type TEE, press the Enter ⏎ key, and the letter "T" will be drawn for us.

To define the procedure, type

TO TEE

Press the Enter ⬅ key.

Notice that Logo now uses > instead of ? as the prompt symbol. This is to remind you that you're defining a procedure and that anything you type will become a part of the procedure. Press the Enter ⬅ key at the end of each line. Type

FD 50
RT 90
BK 20
FD 40

Perhaps you expected the turtle to react immediately to each line of your procedure. But nothing happens. When you're defining a procedure with TO, the turtle does nothing until you finish the definition and type the new command. Type

END

and press the Enter ⬅ key. Logo should respond

TEE DEFINED
?

Notice that Logo again uses ? as the prompt
symbol to let you know you've finished defining
a procedure.

You've just made a new turtle command – TEE.
Try the new command by typing
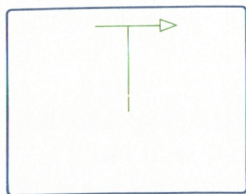
TEE

and pressing the Enter ⬅ key.

🕷 Be sure you clear the screen (CS) before
running the procedure. If nothing happens, or
you see a strange drawing or message on the
screen, you probably have a bug in your
procedure.

TEE

Bug Box
You may make typing mistakes while defining
the procedure. *If the error occurs on the line
you're typing and you spot the error before
pressing the Enter* ⬅ *key* at the end of that
line, correct this error by following the
suggestions in the section "Correcting Typing
Mistakes" in Chapter 2.

*If you notice an error after you've pressed the
Enter* ⬅ *key*, you can't correct it right away.
Finish the definition and make the corrections
when you learn to use the EDIT command in
the next section.

To correct the error now, stop the definition by pressing Ctrl-Break and restart it. (See Chapter 1.) Logo will respond

STOPPED!
?█

The line on which the definition stopped will be printed on the screen.

The ? (prompt symbol) tells you that you are no longer defining a procedure. Now start the procedure again from the beginning and see if the turtle draws the letter T when you type the command TEE.

Defining and Editing Procedures with the Logo Editor

The TO way of defining a procedure is very simple but you can't change any text of the previous lines in the procedure without stopping the definition and retyping it. For this reason, it's generally better to use the Logo editor to define a procedure.

EDIT (or ED) Command

Typing EDIT (or ED) tells Logo that you want to define or edit (make changes). But define or edit what? After typing EDIT type the name of the procedure you want to define or edit. This name must be preceded by a " (quotation mark). Do not type a space between the "(quotation mark) and the name of the procedure.

Let's edit a procedure with the name SQUARE.
Type

🕷EDIT "SQUARE

Press the Enter ⬅ key.

Your turtle drawings and any typed instructions
displayed before on the screen disappear so you
can use the whole screen for text. The words
"LOGO EDITOR" appear at the bottom of the
screen, as you're now using the Logo editor.
Only editing actions will be carried out. Notice
that Logo does not print ? or any prompt
symbol. The title of the procedure
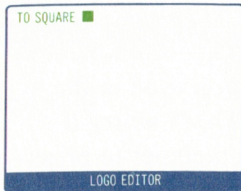
TO SQUARE ■

appears at the top of the screen, with the
Cursor ■ at the end of the title. Press the
Enter ⬅ key.

```
TO SQUARE ■



         LOGO EDITOR
```

Bug Box
Possible Bugs

1. If you don't remember to put a
    " (quotation mark) before **SQUARE** and
    type

    EDIT SQUARE

    Logo will respond

    I DON'T KNOW HOW TO SQUARE

2. If you put a space between " (quotation mark) and SQUARE, the title line will show only

TO ■

3. If you make an error in typing EDIT and spell it "EDET", Logo will respond

I DON'T KNOW HOW TO EDET

If you have any other problems when you're using the Logo editor, press Ctrl-Break. The Break tells Logo to stop editing. Start editing again by typing

EDIT "SQUARE

Now type the commands that you used in the last chapter to draw a square. Don't worry if you make a typing mistake. Remember to press the Enter (←) key after each line.

```
TO SQUARE
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
END
■
                    LOGO EDITOR
```
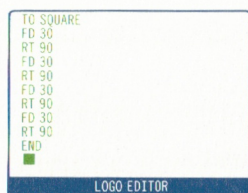
FD 30

RT 90

FD 30

RT 90

FD 30

RT 90

FD 30

RT 90

END
■

Using the Cursor ■ with the Logo Editor

In the Logo editor, you're no longer restricted to making changes before you press the Enter ⏎ key and on the line where the Cursor ■ is located. You can move the Cursor ■ to any place in the text of the procedure in order to correct or change something.

To move the Cursor ■, use the four arrow keys located on the number keypad on the right-hand side of the keyboard. Each key moves the Cursor ■ in the direction of the arrow, one space at a time, unless you hold a key down, in which case the Cursor ■ will continue to move until you release the key. When the Cursor ■ passes over typed characters they remain unchanged.

To move the Cursor ■ forward, press the Cursor Right ⬚ key.

To move The Cursor ■ backward, press the Cursor Left ⬚ key.

To erase a character, move the Cursor ■ on top of that character, press the ⬚ key, and the character hidden by the Cursor ■ will disappear.

If the Cursor ■ is at the end of a line, pressing the ⬚ key will bring the next line of text up to the end of this line. So, in the case of

```
FD 30 ■
RT 90
```

where the Cursor ■ is placed after 30, pressing the (Del) key will result in

FD 30█T 90

Press the Enter (↵) key to separate the lines again.

FD 30
█T 90

If you've made a typing error in the last line, you could move back to the error by using the Cursor Left (←) key. But, it's much easier and faster to use the Cursor Up (↑) key to move the Cursor ■ to where the error is.

Similarly, if you want to move the Cursor ■ down, use the Cursor Down (↓) key.

The Cursor ■ will move only where text is on the screen. If you try to make the cursor go where there is no text, Logo will beep.

Leaving the Editor

The last line in all procedures must be END. This tells Logo that the procedure is complete. However, typing END does *not* get you out of the Logo editor. To get out of the editor, you must press the (Esc) key. If you don't type END, Logo will insert it for you when you press the (Esc) key.

You can easily define more than one procedure while in the Logo editor. As long as you type END when you have finished typing one procedure, you can continue and define another procedure while still in the Logo editor. Type TO followed by the new procedure name to start the definition of the new procedure. When you have finished defining all procedures, press the (Esc) key.

The screen will go blank when you press the (Esc) key, showing that you are out of the Logo editor. In the case of the SQUARE procedure we've been editing, Logo will respond

SQUARE DEFINED
? ■

The ? (prompt symbol) appears on the screen. You are now out of the editor.

Once you've defined a procedure, its name becomes a new turtle command like FD, BK, LT, and RT. So you've just made a new command named SQUARE. To try your new command, type

SQUARE

SQUARE

Using New Commands as Building Blocks

New commands can be used as part of the
definition of other new procedures. This is one
of the strong features of Logo. You can use your
new SQUARE command to create another
design. Type
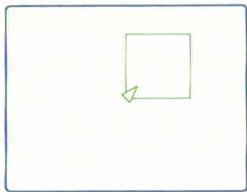
RIGHT 45
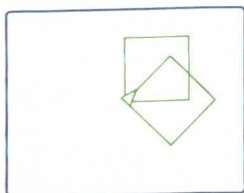
Now type

SQUARE
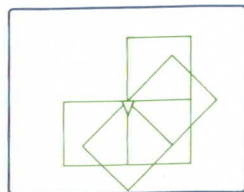
Repeat the commands RT 45 and SQUARE a number of times.

To get the same result without having to type both commands over and over, use the Logo command REPEAT.

REPEAT Command

REPEAT needs two inputs. The first input tells how many times to repeat a command; the second input tells which command you want repeated.

To use REPEAT to get Logo to repeat RT 45 SQUARE three times, type
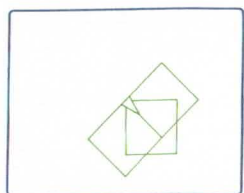
CS
REPEAT 3 [RT 45 SQUARE]

The instructions to be repeated must be put in [ ] (brackets).
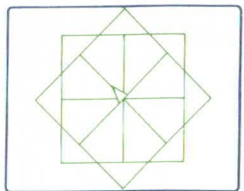
Use REPEAT to perform these commands many times. Type

REPEAT 1000 [RT 45 SQUARE]

The turtle draws a lot of squares which form a design that resembles a star. The turtle will continue to do this for a very long time. To stop the turtle, press Ctrl-Break.

The screen will look like this:

```
STOPPED!:
JUST BEFORE LEAVING SQUARE
? █
```

Bug Box
If Logo is doing anything that you don't understand, press Ctrl-Break. Logo will stop what it's doing and allow you to type another command. If Logo is running a procedure when you press Ctrl-Break, it will tell you where it stopped.

Make a procedure to draw this design. Let's give it the name **SQUARESTAR**. Type

```
EDIT "SQUARESTAR
```

Because you're using the Logo editor, the title will be written on the screen.

```
TO SQUARESTAR █
```

Notice that the Cursor █ is at the end of the title line. Because you don't want to change the title, press the Enter (↵) key, and type

```
REPEAT 8 [RT 45 SQUARE]
END
```

Don't forget to press the (Esc) key when you've finished editing.

It's always a good idea to try out a new procedure to make sure there are no bugs in it. So you can see the effect of the new procedure, put the turtle in the center of the screen heading straight up. Type

CLEARSCREEN (or CS)

Now type

SQUARESTAR
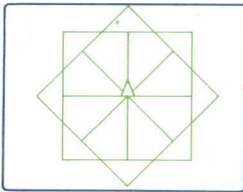
SQUARESTAR

Other Uses of SQUARE

There are many other designs that can use SQUARE. Some examples are:
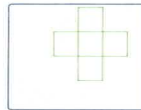
FLAG or   CROSS   FLAGS   MANYFLAGS
FLAGBACK
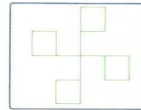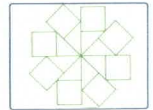
Here are the procedures for these designs:
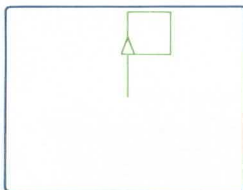
FLAG

TO FLAG
FD 30
SQUARE
END

3-15
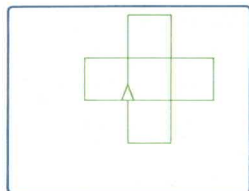
CROSS

```
TO CROSS
REPEAT 4 [FLAG RT 90]
END
```
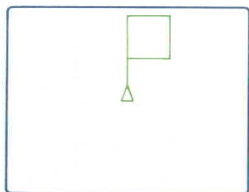

FLAGBACK

```
TO FLAGBACK
FLAG
BK 30
END
```
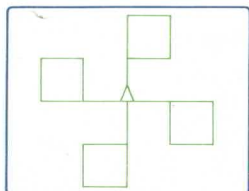

FLAGS

```
TO FLAGS
REPEAT 4 [FLAGBACK RT 90]
END
```
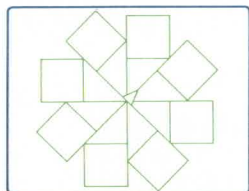

MANYFLAGS

```
TO MANYFLAGS
FLAGS RT 45
FLAGS
END
```

Study these procedures to be sure you understand how they work. Try them yourself. Type each procedure, press the ⎡Esc⎤ key to leave the editor, and then run the procedure.

Notice that in both FLAG and FLAGBACK, the turtle draws the same design but ends up in different states. At the end of both procedures, the turtle has the same heading as it did at the start of each procedure, but at the end of FLAG the turtle is in a different position from the one it had when it began. With FLAGBACK, the turtle moves back to the same position it was in before beginning the procedure.

We can see the effect of these differences in CROSS and FLAGS. CROSS runs FLAG four times, while FLAGS runs FLAGBACK four times.

Note: If you switch the computer off, all the procedures you have written will be erased. See the next chapter for information about how to save your procedures before switching off the computer.

When using the Logo editor, you can only create and edit procedures. To do other actions, you have to leave the Logo editor.

Logo Vocabulary

Here is a list of commands introduced in this chapter, along with their short forms.

Command	Short Form
EDIT	ED
END	
REPEAT	
TO	

Special Keys

Here is a list of special keys introduced in this chapter.

Left Bracket [

Right Bracket]

Ctrl-Break

Cursor Down 〔²↓〕

Cursor Right 〔⁶→〕

Cursor Up 〔⁸↑〕

The name SHAPES (or the name you gave the file) should appear with LF in the next column, followed by a number, a date, and a time. LF means Logo File. Each time Logo saves your procedures, it adds .LF to the end of the name to let you know it is a Logo File. However, when you refer to the file by name, you don't have to use the .LF ending. (See the *Logo Reference* for further information.) The number following the file name refers to the amount of memory used to save the file, and the date and time are records of when you saved the file.

Erasing Files

To erase a file, use the ERASEFILE command.

ERASEFILE Command

The ERASEFILE command takes, as its input, the name of the file you want to erase preceded by a "(quotation mark).

> **Warning:** ERASEFILE erases files forever.

If you type

```
ERASEFILE "SHAPES
```

the SHAPES file will be permanently erased from your diskette.

Replacing Files

To replace an existing file with a new file using the same name, first erase the existing file; then save the new file using the old file name. You may have a file that contains procedures you don't want anymore. Type

```
ERASEFILE "SHAPES
```

to erase the file of **SHAPES** procedures you made before.

Your workspace contains the new procedures that you want to save. Type

```
SAVE "SHAPES
```

This will create a new file of the procedures currently in the workspace.

Another way to replace an existing file is to save your workspace with a new name; then, erase the old file.

Adding to Files

To add one or more procedures to an existing file, bring the file into the workspace and erase the file. Then save all of the procedures presently in the workspace. This file now contains the new procedures plus the old procedures. For example:

```
LOAD "SHAPES
```

loads the **SHAPES** file into your workspace.

```
POTS
```

prints out the procedures in your workspace.

erases your old **SHAPES** file.

SAVE "SHAPES

will create a new file of all the procedures in the workspace.

Use the command **POTS** to check what's in your workspace before using the commands **SAVE** or **ERASEFILE**. If you do this, you will not save procedures you don't want or erase a file by mistake.

Logo Vocabulary

Here is a list of commands introduced in this chapter, along with their short forms.

Command	Short Form
DIR	
ERASE	ER
ERASEFILE	
ERPS	
LOAD	
PO	
POPS	
POTS	
SAVE	
TEXTSCREEN	TS

Chapter 4. Your Workspace

Contents

When you define procedures, Logo puts them in
your workspace. A *workspace* is a space in the
computer memory that lasts only while the
computer is on.

To see what is in your workspace, you can tell
Logo either to print out the titles of all
procedures you've defined or to print out the
definitions of all your procedures.

Printing Out Procedures

Because you have several procedures in your
workspace by now, it's a good idea to make the
whole screen available for text before having
your procedures printed out.

TEXTSCREEN (or TS) Command
To use the entire screen for text, type

TEXTSCREEN (or TS)

POTS Command
POTS, which stands for Print Out TitleS, tells
Logo to print out on the screen the titles of each
of the procedures in the workspace. Type
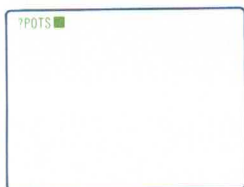
POTS

?POTS■

If your workspace contains the procedures listed
in the last chapter, Logo will respond

```
?POTS
TO MANYFLAGS
TO FLAGS
TO FLAGBACK
TO CROSS
TO FLAG
TO SQUARESTAR
TO SQUARE
TO TEE
? ■
```

POTS

TO MANYFLAGS

TO FLAGS

TO FLAGBACK

TO CROSS

TO FLAG

TO SQUARESTAR

TO SQUARE

TO TEE

? ■

Bug Box
Possible Bugs:

1. If you don't see anything when you type
    POTS, you have no procedures in your
    workspace. You can review the previous
    chapters and retype the ones you want.

2. If you have many procedures in your
    workspace, you may find that the titles go
    by too quickly on the screen and disappear
    as more titles come on. If this happens,
    repeat the command POTS and press
    Ctrl-NumLock as soon as you see the title
    or titles you're looking for. Ctrl-NumLock
    will *pause* whatever is on the screen. Press
    any key to see the rest of the titles.

POPS Command

POPS, which stands for Print Out ProcedureS, prints out on the screen the definitions of all procedures in the workspace. Type

```
?POPS■
```

POPS

Logo responds

```
TO MANYFLAGS
FLAGS
RT 45
FLAGS
END

TO FLAGS
REPEAT 4 [FLAGBACK RT 90]
END

TO FLAGBACK
FLAG
BK 30
END
```

and so on.

PO Command

PO, which stands for Print Out, prints out on the screen the definition of a particular procedure. PO must be followed by the name of the procedure you want. Type

```
?PO "SQUARESTAR■
```

PO "SQUARESTAR

The name of the procedure must be preceded by a " (quotation mark).

4-5

Logo responds

TO SQUARESTAR
REPEAT 8 [RT 45 SQUARE]
END

If you want a printout of more than one
procedure, use PO with a list of procedure
names enclosed in brackets. For example, to
print out the definitions of SQUARE,
SQUARESTAR, and FLAG, type

PO [SQUARE SQUARESTAR FLAG]

Erasing from the Workspace

You can also erase procedures from your
workspace. Be careful: once you erase a
procedure, it no longer exists in your workspace.
If you need it later, you'll have to define it
again.

ERASE (or ER) Command
Several Logo commands can be used to erase
your procedures. The most commonly used is
ERASE (or ER). To erase the procedure TEE,
type

ERASE "TEE

The name of the procedure must be preceded by
a " (quotation mark).

You can also erase a list of procedures by putting the procedure names in brackets. For example, if you wish to erase the procedures CROSS and SQUARESTAR at the same time, type

ERASE [CROSS SQUARESTAR]

ERPS Command

To erase *all* of the procedures from your workspace, you can use the command ERPS, which stands for ERase ProcedureS. Therefore, use ERPS with caution!

Bug Box
Some of the workspace commands like POPS and ERPS can take an additional input. We do not discuss that use in this book. To avoid any problems at this time, we suggest that you use these commands only as described here.

Saving your Procedures

The procedures left in your workspace will disappear as soon as the computer is switched off. However, you can save your procedures permanently on a diskette before you switch the computer off. To do this, you need a properly formatted file diskette. If you don't already have one, see Appendix A and format a diskette now.

If you decide to format now, you won't be able to save the procedures you've made because formatting erases the procedures now in your workspace.

Remove the Logo Language Backup diskette from drive A and replace it with your properly formatted file diskette. Be sure to pull down the tab on the door of drive A.

SAVE Command

The SAVE command saves all procedures now in your workspace. The procedures are saved as one unit or set on your diskette. A set of saved procedures is called a *file*, and each file on your diskette must be given a different name. Choose any name, as long as it has no more than 8 characters. If the name has more than 8 characters, Logo uses only the first 8. For information on the kinds of characters that can be used, see Chapter 4 in the *Logo Reference*. Because all of the procedures now in your workspace draw different kinds of shapes, let's give them the file name SHAPES. Type

```
SAVE "SHAPES
8 PROCEDURES SAVED
?■
```

SAVE "SHAPES

Because the file name is the input, it must be preceded by a ": (quotation mark).

Wait until the red light on drive A is off. Then Logo will tell you how many (*n*) procedures you have saved in this file.

n PROCEDURES SAVED
?■

Bug Box

Possible Bugs:

1. Logo will refuse to execute the SAVE command if a file of the same name already exists on the diskette. The message

    FILE *name* ALREADY EXISTS

    will appear on the screen.

2. Logo will also refuse to execute SAVE if the diskette is unformatted. In this case, the message:

    I'M HAVING TROUBLE WITH THE DISK

    will appear on the screen. Turn to Appendix A to find out how to format a diskette.

The contents of your workspace have now been recorded onto the file diskette under the file name SHAPES. You will now be able to switch the computer off at any time and remove your file diskette. The procedures saved on the diskette can be retrieved the next time you switch the computer on by loading the Logo Language Backup diskette and then loading your file diskette.

Loading Files from a Diskette

To get a file from the diskette and put it into your workspace, use the command LOAD.

LOAD Command

The input for LOAD is the name of the file you want, preceded by a " (quotation mark). For example, type

```
LOAD "SHAPES
? ■
```

LOAD "SHAPES

All of the procedures that were in your workspace when you saved the file SHAPES are now back in your workspace. To see which procedures are there, use the POTS command to print out the procedure titles. Note that the LOAD command does not erase what is in your workspace. If you have procedures in your workspace before you LOAD a file, the procedures loaded are added to your workspace.

Listing Files from a Diskette

To list the names of your files, use the DIR command.

DIR Command

The DIR command checks the names of files already on the diskette. DIR stands for DIRectory. Type

```
DIR
```

The name SHAPES (or the name you gave the file) should appear with LF in the next column, followed by a number, a date, and a time. LF means Logo File. Each time Logo saves your procedures, it adds .LF to the end of the name to let you know it is a Logo File. However, when you refer to the file by name, you don't have to use the .LF ending. (See the *Logo Reference* for further information.) The number following the file name refers to the amount of memory used to save the file, and the date and time are records of when you saved the file.

Erasing Files

To erase a file, use the ERASEFILE command.

ERASEFILE Command

The ERASEFILE command takes, as its input, the name of the file you want to erase preceded by a "(quotation mark).

> **Warning:**   ERASEFILE erases files forever.

If you type

```
ERASEFILE "SHAPES
```

the SHAPES file will be permanently erased from your diskette.

Replacing Files

To replace an existing file with a new file using the same name, first erase the existing file; then save the new file using the old file name. You may have a file that contains procedures you don't want anymore. Type

ERASEFILE "SHAPES

to erase the file of **SHAPES** procedures you made before.

Your workspace contains the new procedures that you want to save. Type

SAVE "SHAPES

This will create a new file of the procedures currently in the workspace.

Another way to replace an existing file is to save your workspace with a new name; then, erase the old file.

Adding to Files

To add one or more procedures to an existing file, bring the file into the workspace and erase the file. Then save all of the procedures presently in the workspace. This file now contains the new procedures plus the old procedures. For example:

LOAD "SHAPES

loads the **SHAPES** file into your workspace.

POTS

prints out the procedures in your workspace.

ERASEFILE "SHAPES

erases your old **SHAPES** file.

SAVE "SHAPES

will create a new file of all the procedures in the workspace.

Use the command **POTS** to check what's in your workspace before using the commands **SAVE** or **ERASEFILE**. If you do this, you will not save procedures you don't want or erase a file by mistake.

Logo Vocabulary

Here is a list of commands introduced in this chapter, along with their short forms.

Command	Short Form
DIR	
ERASE	ER
ERASEFILE	
ERPS	
LOAD	
PO	
POPS	
POTS	
SAVE	
TEXTSCREEN	TS

Chapter 5. The Turtle and Text on the Screen

Contents

When you first start up Logo, before you type any turtle commands, the whole screen can be filled with text. As soon as you type a turtle command, the screen is a large turtle field with six lines at the bottom for text. There are different ways to get the whole screen back for text.

> Note: If you are using two screens (one for text and one for graphics), then the MIXEDSCREEN, FULLSCREEN, TEXTSCREEN, and SETTEXT commands and the F1, F2, and F4 function keys, described in this chapter, will have no effect. The CLEARTEXT command, described in this chapter, will clear the text on your text screen.

MIXEDSCREEN (or MS) Command

```
?REPEAT 5 [PR "HELLO]
HELLO
HELLO
HELLO
HELLO
HELLO
?■
```
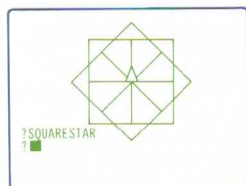
TEXTSCREEN

MIXEDSCREEN

The command TEXTSCREEN (or TS) clears the whole screen for text. You can also press the function key ⎡F1⎤ for TEXTSCREEN. MIXEDSCREEN (or MS) shows the turtle field and the six-line text field. Neither of these commands destroys what was on the two fields. They only change what you can see. Try going back and forth to see what happens. The function key ⎡F2⎤ can be used for MIXEDSCREEN.

FULLSCREEN (or FS)
Command
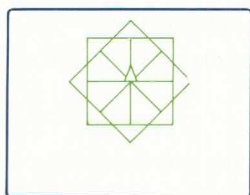
FULLSCREEN

FULLSCREEN (or FS) gives the whole screen to the turtle. No text is visible. After you type FULLSCREEN, the characters that you type will not appear on the screen, but they will be there all the same. You will see this if you type MIXEDSCREEN (MS). You can also press the function key (F4) for FULLSCREEN.

(F1) , (F2) , and (F4) can be pressed while a procedure is running. If you press (F1) while a graphics procedure is running, the textscreen will be shown momentarily.

Try going back and forth between FULLSCREEN and MIXEDSCREEN. Press

(F4)   (F2)

Go back and forth between TEXTSCREEN and MIXEDSCREEN. Press

(F1)   (F2)

When you edit a procedure, the turtle screen is erased; it becomes the edit screen.

CLEARTEXT (or CT) Command

Another useful command is CLEARTEXT (or CT). Whenever there is any text on the screen, CLEARTEXT clears the screen of text and places the cursor at the top of the text portion of the screen.

Changing the Text Portion on the Graphics Screen

When you switch from TEXTSCREEN to the graphics screen, you can use the complete screen for graphics but only six lines at the bottom for text. You can change the portion of the screen that you use for text. The command to do this is SETTEXT.

SETTEXT Command

SETTEXT takes a number as its input. This number tells Logo how many lines of text you want. If you want 13 lines instead of 6, type

SETTEXT 13

The largest number you can use as input for SETTEXT is 25. To set the screen back to 6 lines of text at the bottom, type

SETTEXT 6

Logo Vocabulary

Here is a list of commands introduced in this chapter, along with their short forms.

Command	Short Form
CLEARTEXT	CT
FULLSCREEN	FS
MIXEDSCREEN	MS
SETTEXT	

Special Keys

Here is a list of special keys introduced in this chapter.

[F1]
[F2]
[F4]

Chapter 6. IBM Personal Computer Color Graphics

Contents

The IBM Color/Graphics Monitor Adapter gives you up to 16 colors to use in your turtle graphics.

There are two types of color changes you can make. You can change the color of the whole screen (background color) and the color of the turtle's pen (pen color).

Changing the Background Color

The background colors are number coded as follows:

0	Black	8	Gray
1	Blue	9	Light Blue
2	Green	10	Light Green
3	Cyan	11	Light Cyan
4	Red	12	Light Red
5	Magenta	13	Light Magenta
6	Brown	14	Yellow
7	White	15	High-Intensity White

Think of colors 8 to 15 as lighter versions of colors 0 to 7.

SETBG Command

You can change the background color by using the command SETBG, which stands for SET BackGround.

SETBG takes an input which must be the number of one of the background colors.

Let's try a few of the background colors. Type

```
SETBG 1
SETBG 2
SETBG 3
```
etc.

Notice that the shape of the turtle changes a little with different background colors. SETBG 7 will look different from the others because 7 is white. The white turtle disappears on the white background.

PRINT (or PR) Command

To find out the number of the background color that's on the screen, use the command PRINT (or PR), followed by BACKGROUND or BG. Type

```
PRINT BG
```

Logo responds with

```
3
```

which is the number for the current background color.

PRINT (or PR) is the command that prints on the screen what it receives as an input. Here it receives a number that tells you the background color. It can also be used to print a message on the screen. Type

```
PR [THE BACKGROUND COLOR IS CYAN]
```

Logo responds

```
THE BACKGROUND COLOR IS CYAN
```

Let's define a procedure that changes the background colors one after another. Call it CB, which stands for Change Background.

To see the colors more clearly when Logo runs the CB procedure, use the Logo command WAIT with an input of a number. This will tell Logo how long to wait before running the next command. WAIT 20, for example, makes Logo wait for approximately one second before running the next command. Try different inputs to WAIT. Type

```
TO CB
SETBG 0 WAIT 5
SETBG 1 WAIT 10
SETBG 2 WAIT 15
SETBG 3 WAIT 20
SETBG 4 WAIT 25
SETBG 5 WAIT 30
SETBG 6 WAIT 25
SETBG 7 WAIT 20
SETBG 8 WAIT 15
END
```

6-5

Notice that, in this procedure, we have typed two commands per line. This is a good technique to use with long procedures. It reduces typing time and condenses the procedure on the screen, thereby allowing you to see more of it at the same time.

In this example, we've gone up to background color 8. You could write another procedure to see the background colors 9 through 15.

Now you have a new command called CB. Try running it a couple of times. Type

CB

Instead of typing CB each time to run it, type

REPEAT 3 [CB]

Changing Pen Color

IBM color graphics includes three pen color numbers and two *palettes* of colors. Each palette consists of three pen colors. You have a choice of six pen colors in all. These are the pen color numbers and names:

Pen Color	Palette 0	Palette 1
1	Green	Cyan
2	Red	Magenta
3	Brown	White

When typing the number of pen color you want, you have to check which palette you're using. Logo starts up with Palette 1, which offers the pen colors cyan, magenta, and white.

SETPC Command

The command SETPC, (which stands for SET Pen Color), followed by an input of a number, tells Logo which pen color to use.

To try the pen colors, clear the screen and set the background color to black. Type

```
CS
SETBG 0
SETPC 1
SQUARE
RT 60
SETPC 2
SQUARE
RT 60
SETPC 3
SQUARE
RT 60
```

The pen colors will work as listed here only on a black background. Other background colors may change the pen colors.

SETPAL Command

The command SETPAL (which means SET PALette) tells Logo which of the two palettes to use. SETPAL takes either 0 or 1 as an input.

Because we've been using palette 1, change the palette to 0. Type

```
SETPAL 0
```

Now use the pen colors green, red, and brown.
Type the following commands without clearing
the screen first:

```
SETPC 1
SQUARE
RT 30
SETPC 2
SQUARE
RT 30
SETPC 3
SQUARE
```

PENCOLOR (or PC) Operation
When used with the command PRINT,
PENCOLOR (or PC) gives the number for the
pen color currently on the screen. To try it, type

```
PR PC
```

Because the last SETPC had the input 3, Logo
responds

```
3
```

PALETTE (or PAL) Operation
When used with the command PRINT,
PALETTE (or PAL) gives the current palette
number. Type

```
PR PAL
```

Because the last palette number was 0, Logo
responds

0

To change the pen color back to white, type

SETPAL 1
SETPC 3

Logo Vocabulary

Here is a list of commands and operations
introduced in this chapter, along with their short
forms.

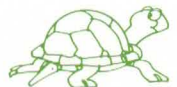| Command	Short Form
PRINT	PR
SETBG	
SETPAL	
SETPC	
WAIT	

Operation	Short Form
BACKGROUND	BG
PALETTE	PAL
PENCOLOR	PC

Chapter 7. More About Editing your Procedures
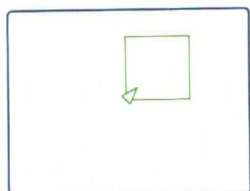
Contents

7-1

The advantage of the Logo editor is that you can define new procedures, change procedures you've already defined, and make editing changes as you define procedures. You may wish to change a procedure to fix a bug or to alter what the procedure does.

Let's define a procedure to draw a diamond. To do this, type

```
TO DIAMOND
SQUARE
RT 45
END
```

Type

```
DIAMOND
```

When you press the Enter ⟨←⟩ key, the turtle draws a square, not a diamond. Why? Because you should have told the turtle to RT 45 *before* the turtle draws a square. To fix this bug, edit the procedure. Type

```
EDIT "DIAMOND
```

The text of the procedure, DIAMOND, is now on the screen.

DIAMOND

```
TO DIAMOND ■
SQUARE
RT 45
END
```

LOGO EDITOR

The Cursor ■ is positioned at the end of the title line, after the last letter D of the word DIAMOND. To edit the procedure DIAMOND, move the Cursor ■ by pressing the Cursor Right ⟨→⟩ key to the beginning of the second line, which will put the Cursor ■ on the S of SQUARE.

7-3

TO DIAMOND
SQUARE
RT 45
END

The ⌈0⌉/Ins key allows you to insert a new line.
Press the ⌈0⌉/Ins key and type

RT 45

When you press the Enter ⌐⌐ key after typing
RT 45, another blank line is inserted into your
procedure, with the Cursor ■ placed at the
beginning of the line.

TO DIAMOND
RT 45
■

SQUARE
RT 45
END

Press the ⌈Del⌉ key to delete the blank line.

TO DIAMOND
RT 45
SQUARE
RT 45
END

Move the Cursor ■ down to the next line by
pressing the Cursor Down ⌈2⌉/↓ key. The
Cursor ■ will then be on the R of the original
RT 45.

Press the [Del] key five times to erase the characters RT 45 on this line.
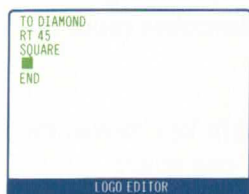
```
TO DIAMOND
RT 45
SQUARE
■
END
```

Now press the [Del] key one more time to delete the blank line.

Leaving the Logo Editor

🕷When you're finished editing, press the [Esc] key. This tells Logo that you've finished editing. Logo prints out a message saying
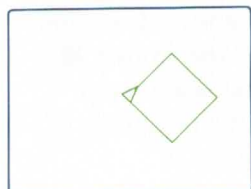
```
DIAMOND DEFINED
?■
```

To try out the procedure, type

```
DIAMOND
```

DIAMOND

> **Bug Box**
> If you are editing and don't like the changes you're making or decide not to make changes after all, press Ctrl-Break. The Break causes Logo to leave the Logo editor and ignore the changes that you made so far. The definition of the procedure will remain as it was before you started editing.

Some Editing Actions

Here is a summary of some useful editing actions. The *Logo Reference* describes more of them.

Cursor Right key moves the Cursor ■ one space forward (to the right).

Cursor Left key moves the Cursor ■ one space backward (to the left).

Cursor Up key moves the Cursor ■ up to the previous line.

Cursor Down key moves the Cursor ■ down to the next line.

Enter key ends a Logo line and moves the Cursor ■ and the text that comes after it to the beginning of the next line.

⬚ key (Tab)	◀— key moves the Cursor ■ to the beginning of the current line. Used with the Shift key.
	—▶ key moves the Cursor ■ to the end of the current line.
⬚ Del	Erases the character directly behind the Cursor ■.
⬚ 0 Ins	Inserts a new line.
⬚ Ctrl ⬚ 6 → Cursor Right	Key combination erases the text from the Cursor ■ to the end of the line.
⬚ Ctrl ⬚ 4 ← Cursor Left	Key combination inserts the line just erased.
⬚ F3	Inserts a copy of last line typed.

Note that all of the above keys except the ⬚ 0 Ins key and the ⬚ key are typematic; that is, if you hold the key down, it will continue executing its function until you release it.

Editing Outside of the Logo Editor

You can use most of the editing key actions we've just described to edit instructions even when you're not in the Logo editor. For example, type
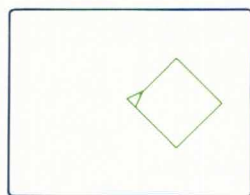
DIAMOND

Press the Enter (←) key.

DIAMOND

Now press the (F3) key. Logo responds by showing what you just typed.

DIAMOND ■

The Cursor ■ is at the end of the line. You can press the Shift key and then the |←— key to move the Cursor ■ to the beginning of the line.

D IAMOND

Type

RT 45
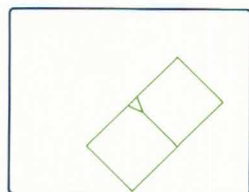
RT 45 DIAMOND

Type a space and press the Enter (←) key again.

Press (F3) and then press the Enter (←) key again.

Try other editing actions both in the Logo editor and outside of it. For more details, see the *Logo Reference*.

Special Keys

Here is a list of special keys introduced in this chapter.

| Ctrl | 4 ← |
| Ctrl | 6 → |
| F3 |
| 0 Ins |
| ← → |
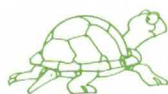
Chapter 8. Drawing a Spider

You learned how to use the procedure named SQUARE to help define a series of new procedures such as FLAG, DIAMOND, etc. We'd now like to show you how to use this same building block approach to draw a spider.
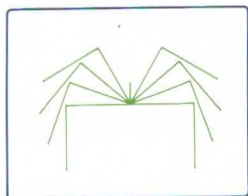
Let's make a spider like the one in the drawing. If you look at it closely, you'll see it has four legs on each side, and that each leg is made by two lines joined at a 90-degree angle.

To draw the spider we can start by making one right leg. Let's use the Logo editor. Type

EDIT "RIGHTLEG

This response will appear at the top of the screen:

TO RIGHTLEG ■

Then type

FD 30
RT 90
FD 30
END

As input to FORWARD, we chose 30. If you want a bigger spider, choose a larger input.

Press the (Esc) key to leave the Logo editor. Clear the screen and type
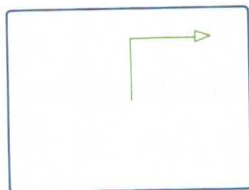
RIGHTLEG

Although this procedure makes a right leg for the spider, the turtle doesn't end up ready to make another right leg. The turtle should be back at the center of the screen where the body of the spider is located.

To make another leg of the spider, we have to get the turtle back to the center of the screen. To do this, use the BACK (or BK) command. We can make this part of the RIGHTLEG procedure by adding three commands to the procedure. Let's do so. Type

```
EDIT "RIGHTLEG
```

The Logo editor shows the procedure you made for RIGHTLEG.

```
TO RIGHTLEG ■
FD 30
FD 90
FD 30
END
```

Now, using the editor keys, add the new commands so your procedure RIGHTLEG reads as follows:

```
TO RIGHTLEG
FD 30
RT 90
FD 30
BK 30        These three new commands
LT 90        return the turtle to where it was
BK 30■       at the start of RIGHTLEG.
END
```

Press the (Esc) key to leave the editor. Clear the screen and run RIGHTLEG again.
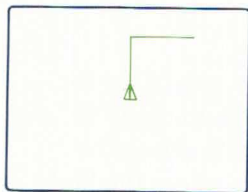
```
RIGHTLEG
```

There's still a problem with this leg: it's stuck up in the air.

RIGHTLEG

To get it down on the ground, type

```
CS
RT 90
RIGHTLEG
```

To make the next right leg, the turtle has to turn a little to the left before it draws **RIGHTLEG** again. Let's try that.
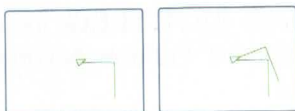
```
LT 20
RIGHTLEG
```

You can experiment with different angles as input for **LEFT**.

Good! Continue repeating the instructions

```
LT 20
RIGHTLEG
```

until the turtle has drawn four legs. Now make a procedure using this plan and call the new procedure **RIGHTSIDE**. Type

```
TO RIGHTSIDE
RT 90
REPEAT 4 [RIGHTLEG LT 20]
LT 10
END
```

Remember to put [ ] (brackets) around the instructions to be repeated.

8-5
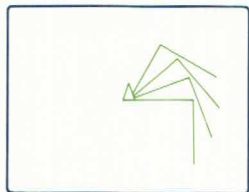
RIGHTSIDE

To try out this procedure, clear the screen, and type

```
RIGHTSIDE
```

Notice that the command LT 10 returns the turtle to the same position and heading it was in at the start of the procedure. It is good practice to always leave the turtle in the state in which you found it.

Now that we have finished the right legs, let's work on the left legs.

Let's first define LEFTLEG, which will be the same as RIGHTLEG, except that the RT commands will now become LT commands. Type

```
TO LEFTLEG
FD 30
LT 90
FD 30
BK 30
RT 90
BK 30
END
```

Run LEFTLEG to make sure it has no bugs in it. Now use LEFTLEG to write LEFTSIDE. Type
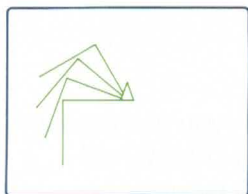
LEFTSIDE

```
TO LEFTSIDE
LT 90
REPEAT 4 [LEFTLEG RT 20]
RT 10
END
```

Try LEFTSIDE.

Finally, put RIGHTSIDE and LEFTSIDE together to draw the whole spider. Type

```
TO SPIDER
LEFTSIDE
RIGHTSIDE
FD 10
BK 10
END
```

The procedures LEFTSIDE and RIGHTSIDE are used as commands in the definition of the SPIDER procedure. We can think of SPIDER as the *superprocedure* and LEFTSIDE and RIGHTSIDE as *subprocedures*.
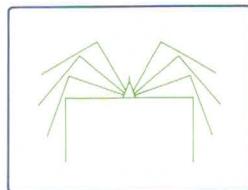
SPIDER

To see the finished spider, clear the screen, and type

```
SPIDER
```

Notice that the turtle is still on the screen – on the spider's back. To remove the turtle, type
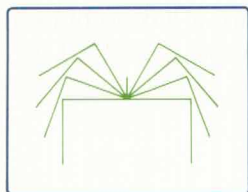
HIDETURTLE (or HT)

To bring the turtle back before you continue, type

SHOWTURTLE (or ST)

Here are some other designs you can produce on the screen by using RIGHTLEG and LEFTLEG to make up new procedures.
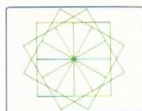
MAN   SWIRL   SPINSTAR

Remember that the procedures in your workspace will disappear as soon as you switch the computer off. You can save them permanently on your file diskette by using the SAVE command. See Chapter 4 for details.

The turtle can draw different triangle shapes. The triangle you can start working with has equal sides and equal angles.

Let's make the turtle take 30 steps forward. Type

FD 30

Now comes the big question. How many degrees does the turtle have to turn to draw this triangle? In geometry, we learn that triangles with three equal sides have 60-degree angles. Look what happens when the turtle turns 60 degrees. Type

RT 60
FD 30
RT 60

```
FD 30
RT 60
```

Interesting, but it's not a triangle! Finish it if
you want to find out what kind of figure you'll
make.

The turtle didn't turn enough at each angle.
How much more should it turn? Let's think
about the turtle's trip along the *outer* edges of a
triangle. The turtle must turn a total of 360
degrees (a complete circle) before it returns to
its starting state. There are three turns in a
triangle: 360 divided by 3 is 120. The turtle has
to turn 120 degrees at each angle!

Now draw the triangle using 120 as the input for
RIGHT. Type


```
CS
FD 30
RT 120
FD 30
```

Chapter 9. Triangles

Contents

The turtle can draw different triangle shapes.
The triangle you can start working with has
equal sides and equal angles.

Let's make the turtle take 30 steps forward.
Type

FD 30

Now comes the big question. How many
degrees does the turtle have to turn to draw this
triangle? In geometry, we learn that triangles
with three equal sides have 60-degree angles.
Look what happens when the turtle turns 60
degrees. Type

RT 60
FD 30
RT 60

```
FD 30
RT 60
```

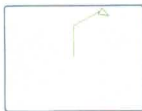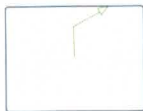Interesting, but it's not a triangle! Finish it if you want to find out what kind of figure you'll make.

The turtle didn't turn enough at each angle. How much more should it turn? Let's think about the turtle's trip along the *outer* edges of a triangle. The turtle must turn a total of 360 degrees (a complete circle) before it returns to its starting state. There are three turns in a triangle: 360 divided by 3 is 120. The turtle has to turn 120 degrees at each angle!

Now draw the triangle using 120 as the input for **RIGHT**. Type


```
CS
FD 30
RT 120
FD 30
```

```
RT 120
FD 30
RT 120
```

Now define the procedure **TRIANGLE**. Let's use the editor to do this. Type

```
EDIT "TRIANGLE
```

and now type

```
REPEAT 3 [FD 30 RT 120]
END
```


TRIANGLE

Try it out.
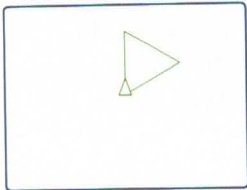
Play with this procedure a bit. For example, type

```
REPEAT 3 [TRIANGLE RT 120]
```
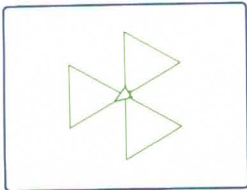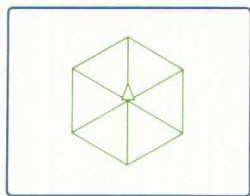
REPEAT 6 [TRIANGLE RT 60]
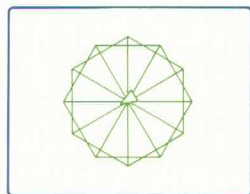
REPEAT 100 [TRIANGLE RT 30]

In this last example, the turtle retraces its path many times. You can always stop the turtle by pressing Ctrl-Break.

You may want to figure out how many times the turtle needs to repeat a certain set of commands. In the previous figure the turtle turns 30 degrees each time, so it has to repeat the set of commands 12 times (360 divided by 30) in order to complete the pattern. Logo can do arithmetic so it can divide 360 by 30 for you. To tell it to do this, type 360/30 as shown.

REPEAT 360/30 [TRIANGLE RT 30]

Here are some other designs you can make using **TRIANGLE**.
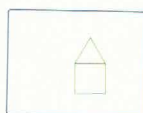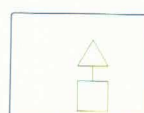
TENT   TREE   HOUSE   WELL

Let's make the turtle draw TENT. Running
TRIANGLE is not enough. The TENT will be
tipped. Type

```
CS
TRIANGLE
```

Try turning the turtle RT 90 first and then run
TRIANGLE. This time the tent is upside down.
Type
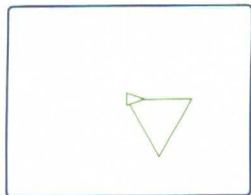
```
CS
RT 90
TRIANGLE
```

Because we have turned the tent too far to the
right, we must turn it back to the left the amount
of one of its angles. We learned earlier that each
inside angle of a triangle with equal sides is 60
degrees. If we now turn the turtle RT 90 and
then LT 60, the turtle is set up for drawing a
tent. Of course, we could just turn the turtle
RT 30. Type
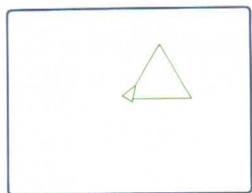
```
CS
RT 30
TRIANGLE
```

Type in the procedure.

```
TO TENT
RT 30
TRIANGLE
END
```

TENT into TREE

Now use TENT to help draw TREE. Type


```
CS
TENT
RT 60
FD 15
```



```
RT 90
FD 15
RT 180
```

Define TREE, using TO or EDIT, by typing in
the above commands except for CS. Now, make
three or four trees appear on the screen by
typing


```
TREE
RT 90
PU
FD 30
```
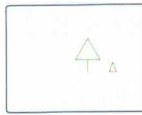
```
LT 90
PD
TREE
```

It is a good habit to put the setup instructions in a separate procedure. SETTREE will set up the turtle to draw a new tree on the screen. Let's define a procedure for SETTREE. Type

```
TO SETTREE
RT 90
PU
FD 30
LT 90
PD
END
```

You can now use **TREE** and **SETTREE** as many times as you like. Type

```
REPEAT 3 [TREE SETTREE]
```
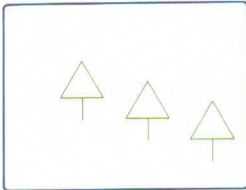

To change the distance between trees, edit **SETTREE** and increase the amount the turtle goes forward.

Turtle Makes a House

You have taught the turtle to make a square and a triangle. Now put them together to make a house. Type
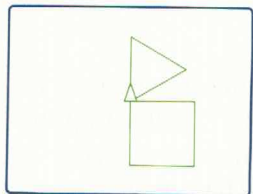
```
SQUARE
FD 30
TRIANGLE
```

With FD 30 you can put the triangle on top of the square.

They're now together but you still don't have a house; your roof is tipped. To fix this bug, use **TENT** instead of **TRIANGLE** so the roof sits flat on the house. Type

```
SQUARE
FD 30
TENT
```
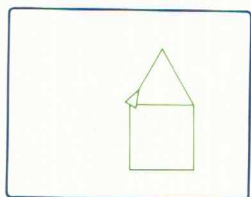
HOUSE
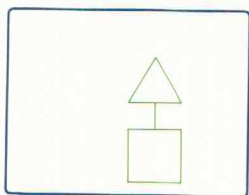
Now use some of the shapes we've just made to draw the well.

WELL

Chapter 10. Variables: Big Squares and Small Squares

Contents

In Chapter 3, we defined a procedure, called SQUARE, having sides of 30 steps. You may, however, want the turtle to draw squares with sides of 60, 50, 100, 10, and so on. One way of doing this is to have many procedures: SQUARE60, SQUARE50, SQUARE100, etc. But there's a shorter way that's even better. We can change SQUARE so that it takes an input. Then tell SQUARE how long to make its sides by typing SQUARE followed by a number input, for example, SQUARE 50, SQUARE 40, and SQUARE 30.

Let's make a procedure for drawing different sized squares. BOX may be a good name because it reminds us of squares. Now we'll do something new: we'll call it BOXR (R for right). This makes us think of a square turning to the right. We could also define a square which turns to the left and call it BOXL (L for left).

A shorter way to type the definition of BOXR is to change SQUARE in the editor. If we change the name of the procedure before we leave the editor, we will not change the definition of SQUARE. Type

EDIT "SQUARE

If your SQUARE procedure is in your workspace, Logo prints

```
TO SQUARE ■

FD 30

RT 90

FD 30

RT 90

FD 30

RT 90

FD 30

RT 90

END
```

TO SQUARE ■
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
END

LOGO EDITOR

10-3

First, let's change the name of the procedure from **SQUARE** to **BOXR**. The Cursor ■ is at the end of the line, after the E of **SQUARE**. Move it to the letter S of **SQUARE**. Use Cursor Left ⌊⁴←⌋ .

```
TO ▊QUARE
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
END
          LOGO EDITOR
```

TO ▊QUARE

```
TO ■
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
END
          LOGO EDITOR
```

Now, erase the word **SQUARE** using the Ctrl ⌊⁶→⌋ (delete to the end of the line) and type the word **BOXR**.

TO BOXR

The Logo editor now shows
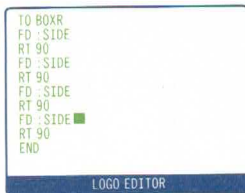
```
TO BOXR■
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
FD 30
RT 90
END
          LOGO EDITOR
```

TO BOXR ■

FD 30

RT 90

FD 30

RT 90

FD 30

RT 90

FD 30

RT 90

END

Let's change **BOXR** so it takes an input like **FORWARD** does. The procedure will let us draw squares of various sizes. How do we tell Logo to do this?

The first instruction has to be FORWARD some amount. When we are writing the procedure, we don't know what this amount will be. It can be different each time we run BOXR. We handle this situation by giving a name to the amount. Let's call it SIDE. Type FD :SIDE in place of each FD 30. This means FORWARD whatever number happens to be in SIDE. This will be an input when you run BOXR. The character : (colon) tells Logo that the word that follows it is an input that can vary each time the procedure is run. This is called a *variable*. It is a container that can have in it a number, another word, a list of words, or a list of lists. We will discuss inputs later on. Replace each 30 with :SIDE so you have this on the screen:
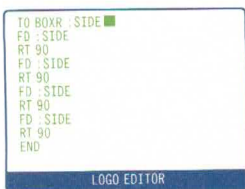
```
TO BOXR
FD :SIDE
RT 90
FD :SIDE
RT 90
FD :SIDE
RT 90
FD :SIDE
RT 90
END
```

```
TO BOXR
FD :SIDE
RT 90
FD :SIDE
RT 90
FD :SIDE
RT 90
FD :SIDE
RT 90
END
```

LOGO EDITOR

One more thing is needed to turn this into a procedure. To indicate that BOXR needs an input and that this input will be called SIDE, edit the title line to look like this:

```
TO BOXR :SIDE
FD :SIDE
RT 90
FD :SIDE
RT 90
FD :SIDE
RT 90
FD :SIDE
RT 90
END
```

LOGO EDITOR

```
TO BOXR :SIDE
```

Now press [Esc] to leave the editor.

10-5

When we use the command **BOXR**, we have to follow it with an input like this:

BOXR 20

This is a box whose sides are 20 steps.
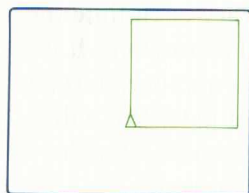
BOXR 20

BOXR 100

This is a box whose sides are 100 steps.

BOXR 100

BOXR now makes the turtle draw a square of any size depending on the number you give it as an input.
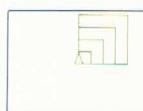
BOXR 10
BOXR 20
BOXR 30
BOXR 40

When defining this procedure, you wanted to tell the turtle how to draw a square but you didn't know what size square you might need. Indeed, you wanted to be able to draw squares of all possible sizes. When you were ready to type the FORWARD command, you knew that FORWARD needed an input. To give the input a name, we called it SIDE. In Logo, the expression :SIDE means "whatever happens to be in the container called :SIDE." If Logo is to carry out the command FORWARD :SIDE, there must be something in the container.

The container is filled when you use BOXR followed by a number input, as in BOXR 10 or BOXR 15. When Logo obeys that command, 10, 15, or whatever you typed as the input is put into the container named SIDE. BOXR can then look into the container at a later time.

Bug Box
Possible Bugs:

1. You typed a space between : (colon) and SIDE.

2. You forgot to use : (colon).

3. You typed :SISE or made some other mistake so that it is different from the input written on the title line.

4. You inserted an extra instruction in BOXR.

5. You accidentally erased an instruction in BOXR.

6. You typed a : (colon) in front of a number.

Using BOXR

Here are some examples of drawings you can
make by using the **BOXR** command.
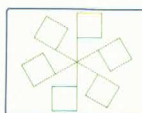
| SQUARES | DIAMONDS | 6FLAG 30 | SPINFLAG 30 |
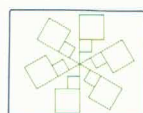
Try some of these procedures to see how you
could use **BOXR**.
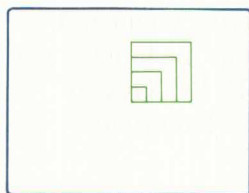
SQUARES

```
TO SQUARES
BOXR 10
BOXR 20
BOXR 30
BOXR 40
END
```
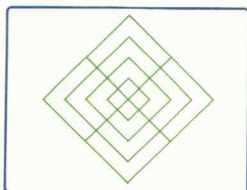

DIAMONDS

```
TO DIAMONDS
RT 45
REPEAT 4 [SQUARES RT 90]
END
```
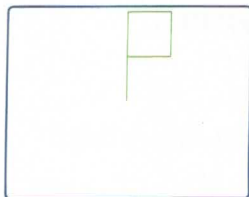
FLAGR 30

```
TO FLAGR :SIZE
FD :SIZE
BOXR :SIZE
BK :SIZE
END
```
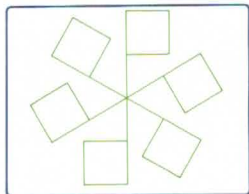

6FLAG 30

```
TO 6FLAG :SIZE
REPEAT 6 [FLAGR :SIZE RT 60]
END
```
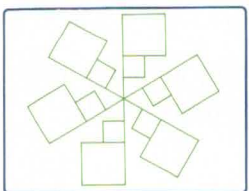

SPINFLAG 30

```
TO SPINFLAG :SIZE
6FLAG :SIZE
6FLAG :SIZE / 2
END
```

Being able to control the size of a shape makes that procedure much more useful and interesting.

Big Triangles and Small Triangles

We can also define a triangle procedure that takes an input. Type

`ED "TRIANGLE`

Logo prints

```
TO TRIANGLE ■
REPEAT 3 [FD 30 RT 120]
END
```

LOGO EDITOR

```
TO TRIANGLE ■
REPEAT 3 [FD 30 RT 120]
END
```

Now change **TRIANGLE** to take an input for the size of its sides. Change the name of the procedure to **TRIANGLER**.

```
TO TRIANGLER :SIDE
REPEAT 3 [FD :SIDE ■ RT 120]
END
```

LOGO EDITOR

```
TO TRIANGLER :SIDE
REPEAT 3 [FD :SIDE ■ RT 120]
END
```

Use **TRIANGLER** to make these designs. Look at your old **TREE** procedure for help.
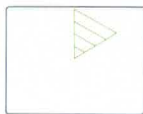
TRIANGLES **TRISTAR** **TREES**

This is one way to make each design:
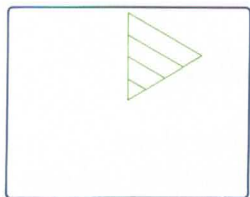
TRIANGLES

```
TO TRIANGLES
TRIANGLER 10
TRIANGLER 20
TRIANGLER 30
TRIANGLER 40
END
```
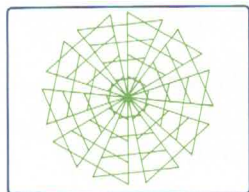

TRISTAR

```
TO TRISTAR
REPEAT 10 [TRIANGLES RT 36]
END
```
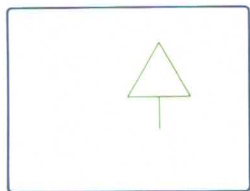

TREER 30

```
TO TREER :SIDE
RT 30
TRIANGLER :SIDE
RT 60
FD :SIDE / 2
LT 90
BK :SIDE / 2
END
```
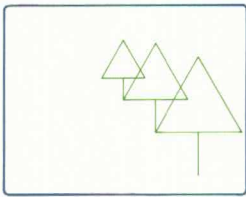
```
TO TREES
TREER 30
TREER 40
TREER 50
END
```

TREES

Arithmetic Operations

As you may have noticed in some of the
examples above, you can do arithmetic in Logo.
For example,

PRINT 5 + 3

The + (plus) is used by Logo for addition.

PRINT tells Logo to print the result on the
screen.

Logo types

8

PRINT 4 * 23

The * (asterisk) is used by Logo for
multiplication.

Logo types

92

PRINT 345 - 32

The — (minus) is used by Logo for subtraction.

Logo types

313

PRINT 25 / 5

The / (slash) is

Logo types

5

Logo computes multiplication and division
before addition and subtraction. In this example,
3 * 6 is computed first (18) and the 5 is added
to the result (18 + 5 = 23).

PRINT 5 + 3 * 6

You can give Logo two operations to compute.

Logo types

23

For more discussion about arithmetic in Logo,
consult the *Logo Reference*.

Logo Vocabulary

Here is a list of special characters and operations introduced in this chapter.

The : (colon) informs Logo that the word that follows it is a variable that can contain a number, another word, a list of words, or a list of lists.

Arithmetic Operations

/   division
−   subtraction
+   addition
*   multiplication
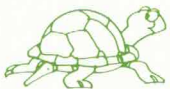
Chapter 11. Procedures with More than One Input: Rectangles and Polygons

Contents

In the last chapter, we saw how to make squares and triangles of any size by letting procedures take an input. However, we can go much further than that and make many interesting designs by defining procedures that take more than one input.

Rectangles

To make rectangles of different sizes, all you need to do is to define a procedure with two inputs. Each input will give the length of each side. Type in the following procedure:

```
TO RECTANGLE :HEIGHT :WIDTH
FD :HEIGHT
RT 90
FD :WIDTH
RT 90
FD :HEIGHT
RT 90
FD :WIDTH
RT 90
END
```

You use it in exactly the same way as procedures with a single input, except you must give two numbers instead of just one. Try


```
RECTANGLE 50 20
RECTANGLE 20 50
```

Polygons

Just as you can change the number of steps
the turtle takes by using an input, you can
also change how much it turns. In fact, you
can get some really beautiful and surprising
patterns by changing both of these two parts
of the turtle's state. The following procedure
takes two inputs: one stating the number of steps
the turtle should take and one stating how much
it should turn. Define **POLY** as follows:

```
TO POLY :STEP :ANGLE
FD :STEP
RT :ANGLE
POLY :STEP :ANGLE                    (Recursive call)
END
```

Now try it!
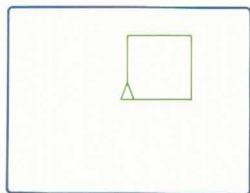
```
POLY 30 90
```


POLY 30 90

The turtle does not stop because **POLY** keeps
telling it to run **POLY** again, go forward, and
turn. The procedure keeps starting over again; it
runs in a loop. To stop **POLY** and the turtle,
press Ctrl-Break. Logo responds

```
STOPPED! IN POLY:
FD :STEP
? ■
```

Here, **POLY** stopped on the **FD :STEP** line.

When a procedure runs in a loop by calling
itself as **POLY** does on the fourth line of the
procedure, we say that it is *recursive*. We'll
learn more about recursive procedures in
Chapter 17.

Here are some other suggestions for exploring POLY. Try your own inputs, too. Remember to clear the screen (CS) before you try each new design.
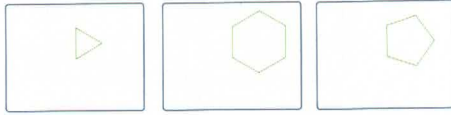
POLY 30 120
POLY 30 60
POLY 30 72
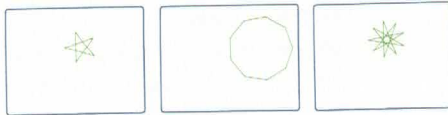
POLY 30 144
POLY 30 40
POLY 30 160

Adding Inputs

You can have as many inputs as you want in a procedure. Just remember to put the inputs beside each other on the title line and to place a : (colon) before each input. Remember to leave a space between each input. Here is a procedure that draws patterns that look like sun rays. Type

```
TO SUN :FDSTEP :BKSTEP :TURN
FD :FDSTEP
BK :BKSTEP
RT :TURN
SUN :FDSTEP :BKSTEP :TURN        (Recursive call)
END
```

11-5

Notice that this procedure is also recursive; it will go on and on until you press Ctrl-Break to stop it. Try
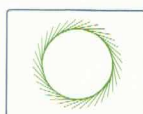
🕷 SUN 70 60 20
SUN 50 40 10

> **Bug Box**
> If your drawings don't look like the pictures shown here, check to see if the three inputs are placed in the same order, in the line that calls SUN again, as they are in the title line. If the order changes, the design will also change.

Chapter 12. Circles and Arcs

Contents
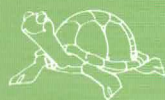
So far, we've seen the turtle make straight line designs. But the turtle can do much more than that. It can even draw circles because it is able to make curves. Curves are made by taking a very small step and turning the direction of the turtle just a little bit. By taking many small steps and many small turns, you'll end up with a whole circle.

To make a complete circle, the turtle has to turn a total of 360 degrees. Type

```
REPEAT 360 [FD 1 RT 1]
```
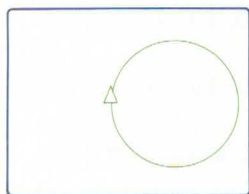
You will notice that the circle takes a long time to draw. That's because the turtle needs to repeat the two instructions 360 times (as many times as drawing 90 squares). You can draw the circles a little faster if you hide the turtle (HT).

You can change the size of the circle in a number of ways. One way is to change the number of steps the turtle moves forward.

```
TO CIRCLE :STEP
REPEAT 360 [FD :STEP RT 1]
END
```

The input for STEP must be quite small. Try the following numbers:
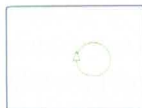
```
CIRCLE .5
CIRCLE 1.5
```

This is a big circle that goes off the screen.

You can also get circles with certain inputs for STEP and ANGLE in the POLY procedure. Try
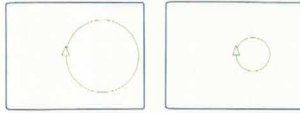

```
POLY 1 1
POLY 1 2
```

POLY 1 1 is the same as CIRCLE 1

POLY 1 2 is the same as CIRCLE .5

Remember to press Ctrl-Break to stop POLY when the design is complete. Try increasing and decreasing the angle. See if you can find out when the drawing produced by POLY stops being a circle.

Circle with Radius

With the CIRCLE procedure we drew circles by repeating 360 times whatever STEP we wanted to use. Sometimes we wish to choose the size of a circle by its radius (the distance from the center to a point on the circle). We can write another procedure with the radius as input that uses CIRCLE as a subprocedure. In this subprocedure, Logo calculates STEP for us. Let's call the superprocedure CIRCLER (R for right).

```
TO CIRCLER :RADIUS
CIRCLE 2*3.14*:RADIUS/360
END
```

Let's hide the turtle (HT) and try


```
CIRCLER 20
CIRCLER 10
```

We can define a left-turn circle by the following procedures:

```
TO CIRCLE1 :STEP
REPEAT 360 [FD :STEP LT 1]
END

TO CIRCLEL :RADIUS
CIRCLE1 2*:RADIUS*3.14/360
END
```

Try
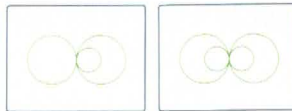

```
CIRCLEL 20
CIRCLEL 10
```

Projects Using Circles

Here are some projects using circles. Try them.
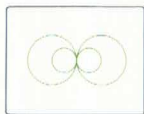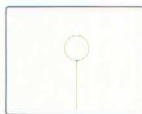
EYES   BALLOON   LOLLIPOP   FLOWER

The procedure EYES uses two subprocedures, REYE and LEYE in its definition.
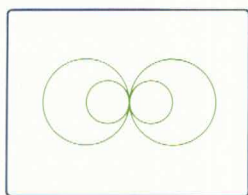
EYES

```
TO EYES
REYE
LEYE
END
```
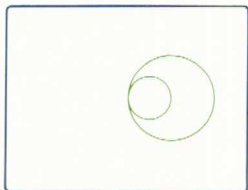

REYE

```
TO REYE
CIRCLER 20
CIRCLER 10
END
```
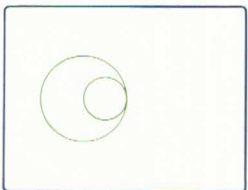

LEYE

```
TO LEYE
CIRCLEL 20
CIRCLEL 10
END
```

12-6
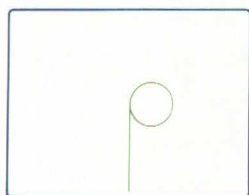
```
TO BALLOON
FD 40
CIRCLER 10
END
```

BALLOON
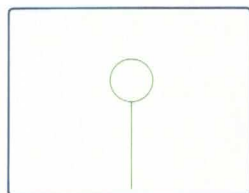

```
TO LOLLIPOP
FD 40
LT 90
CIRCLER 10
END
```
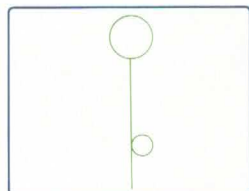
LOLLIPOP


```
TO FLOWER
FD 20
CIRCLER 5
LOLLIPOP
END
```

FLOWER

In the next example, we define FACE with an input. It uses several subprocedures: HEAD, EYES1, MOUTH, and NOSE. Notice that the value of the input, SIZE, is changed for some of these procedures. When FACE calls EYES1, MOUTH, and NOSE, the input that is given to each of these procedures is the original SIZE divided by 5. Therefore, when these three procedures use SIZE, they use this new value of SIZE. Try FACE and see if you can keep track of the value of SIZE in each part.

12-7

FACE 30

```
TO FACE :SIZE
HEAD :SIZE
EYES1 :SIZE / 5
MOUTH :SIZE / 5
NOSE :SIZE / 5
END
```
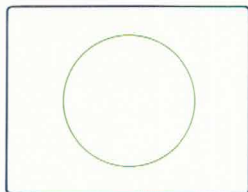

HEAD 30

```
TO HEAD :SIZE
PU
FD :SIZE
RT 90
PD
CIRCLER :SIZE
PU
LT 90
BK :SIZE
PD
END
```
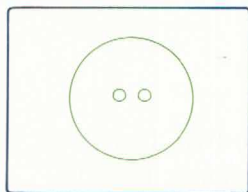

EYES1 6

```
TO EYES1 :SIZE
LEYE1 :SIZE
REYE1 :SIZE
END

TO LEYE1 :SIZE
PU
LT 90
FD :SIZE
PD
CIRCLER :SIZE / 2
PU
BK :SIZE
RT 90
PD
END
```

12-8

```
TO REYE1 :SIZE
PU
RT 90
FD :SIZE
PD
CIRCLEL :SIZE / 2
PU
BK :SIZE
LT 90
PD
END
```
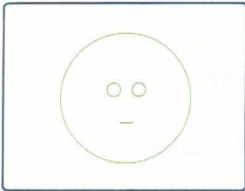

MOUTH 6

```
TO MOUTH :SIZE
PU
BK 2 * :SIZE
RT 90
FD :SIZE / 2
PD
BK :SIZE
PU
FD :SIZE / 2
LT 90
FD 2 * :SIZE
PD
END
```
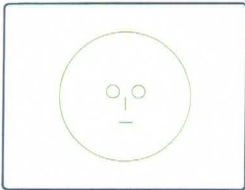

NOSE 6

```
TO NOSE :SIZE
BK :SIZE
HT
END
```

Remember to type **SHOWTURTLE** to get the turtle back.
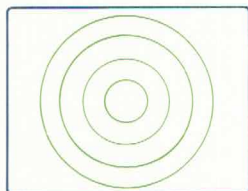
BULLSEYE

```
TO BULLSEYE
HEAD 10
HEAD 20
HEAD 30
HEAD 40
END
```

Drawing Arcs

We said earlier that circles are made by taking many small steps and making many small turns. By limiting the total number of steps and turns, you can draw a piece of a circle called an *arc*. If you change your CIRCLE procedure to repeat only 180 times instead of 360, you will get a semicircle, which is one type of arc. Define a semicircle.

```
TO SEMICIRCLE :STEP
REPEAT 180 [FD :STEP RT 1]
END
```
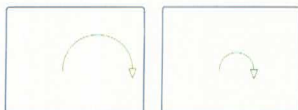


```
SEMICIRCLE 1
SEMICIRCLE .5
```

Repeating only 90 times, you will get a quarter of a circle. Try

```
TO QCIRCLE :STEP
REPEAT 90 [FD :STEP RT 1]
END
```



```
QCIRCLE 1
QCIRCLE 1.5
```

To define a general procedure for drawing arcs, you need only to add a second input to specify the number of degrees you want to turn in the arc. This, of course, determines the total length of the arc itself. Type

```
TO ARC :STEP :DEGREES
REPEAT :DEGREES [FD :STEP RT 1]
END
```

Now try some different inputs.
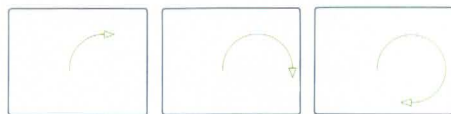

```
ARC 1 90
ARC 1 180
ARC 1 270
```

Arc with Radius

It's useful to have procedures to draw right-turn
arcs and left-turn arcs of a given radius.

They are:

```
TO ARCR :RADIUS :DEGREES
ARC 2*:RADIUS*3.14/360 :DEGREES
END

TO ARCL :RADIUS :DEGREES
ARC1 2*:RADIUS*3.14/360 :DEGREES
END

TO ARC1 :STEP :DEGREES
REPEAT :DEGREES [FD :STEP LT 1]
END
```

These procedures take two inputs: the first input
is the radius of the circle from which the arc is
taken (the distance from the center to the edge
of the circle) and the second input is the degrees
of the arc (the length of the edge). Try


```
ARCR 30 90
ARCR 30 180
```

Clear the screen and try some other inputs.
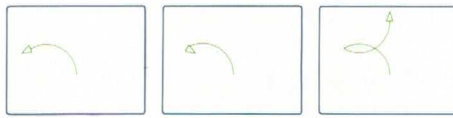
```
ARCL 40 120
LT 120
ARCL 40 120
```

A fish!
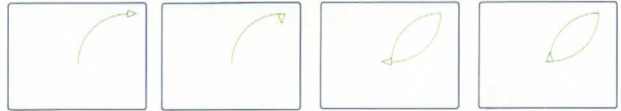

```
ARCR 40 90
RT 90
ARCR 40 90
RT 90
```

A petal!

Arcs, Petals, Flowers, and Swans

Write a general procedure to draw petals like the one we just drew.

```
TO PETAL :SIZE
ARCR':SIZE 90
RT 90
ARCR :SIZE 90
RT 90
END
```

Try

PETAL 40
PETAL 30

PETAL 40
PETAL 30

REPEAT 8 [PETAL 40 PETAL 30 RT 45]
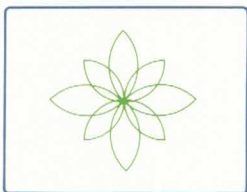
REPEAT 4 [PETAL 30 RT 45 PETAL 40 RT 4→
5]

Logo lines can continue beyond the single lines
you see on the screen. If you type in a screen
line of more than 38 characters, Logo continues
by putting a → (Forward Arrow) in place of the
last character of the line. The rest of the text
flows onto the next screen line, and is
considered as being on the same Logo line. If
you are at top level, you cannot have more than
128 characters in one Logo line. If you try to
exceed this the computer will beep.

You may want to make a FLOWER out of one
of these designs. Now let's draw a SWAN.
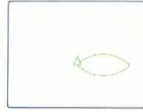
SWAN   BODY   NECK   HEAD1

SWAN is made up of three parts, so we will
have one procedure for each of them. There are
really only two different shapes: PETAL and
NECK. Use PETAL to make BODY and
HEAD. Notice again that the value of SIZE is
changed for different procedures.
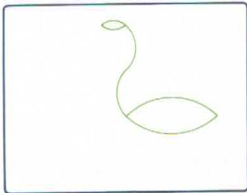
SWAN

```
TO SWAN :SIZE
BODY :SIZE
NECK :SIZE / 2
HEAD1 :SIZE / 4
HT
END
```

It's a good idea to do a HIDETURTLE so the
turtle doesn't show in the design.
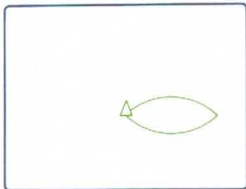
BODY 60

```
TO BODY :SIZE
RT 45
PETAL :SIZE
LT 45
END
```
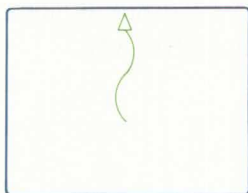
```
TO NECK :SIZE
LT 45
ARCR :SIZE 90
ARCL :SIZE 90
RT 45
END
```

```
TO HEAD1 :SIZE
LT 135
PETAL :SIZE
RT 135
END
```

Now type
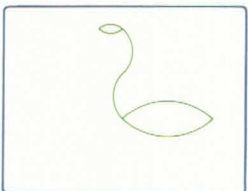
```
SWAN 60
```

Remember to do a **SHOWTURTLE** before going on to another design.

Chapter 13. Spirals
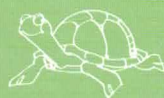
The **POLY** procedure makes the turtle draw
closed figures that finish off with the turtle trail
ending exactly where it started. An exception
occurs when the turtle turns 0 or 360 degrees or
a multiple of 360 on each round. Then, it walks
in a straight line. An example is POLY 1 360.
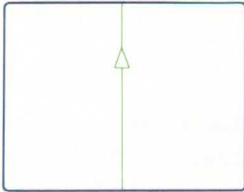
```
TO POLY :STEP :ANGLE
FD :STEP
RT :ANGLE
POLY :STEP :ANGLE
END
```

POLY 1 360

The turtle draws closed figures because it goes
forward, rotates a fixed amount, and eventually
gets back to where it started. We can easily
have the turtle draw a spiral by increasing its
forward step on each round of the procedure so
it never gets back to where it started. Let's
change **POLY** and name it **SPI**. That way we
can make a spiral drawing procedure. Type

```
EDIT "POLY
```

Now the Logo editor shows the **POLY**
procedure.

```
TO POLY :STEP :ANGLE ■
FD :STEP
RT :ANGLE
POLY :STEP :ANGLE
END
```

13-3

Change the name of the procedure from POLY to SPI.


```
TO SPI :STEP :ANGLE
FD :STEP
RT :ANGLE
SP▮ :STEP :ANGLE
END
```

We must make one more change to the recursion line (the line that repeats the first line). We tell SPI to add 2 steps to :STEP. Thus,


```
TO SPI :STEP :ANGLE
FD :STEP
RT :ANGLE
SPI :STEP +▮ :ANGLE
END
```

Remember to press the [Esc] key to leave the editor.

Now try SPI.
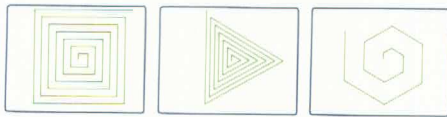

```
SPI 5 90
SPI 5 120
SPI 5 60
```
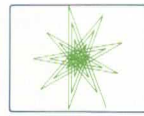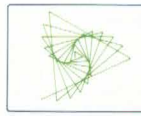
```
SPI 5 144
SPI 5 125
SPI 5 160
```

Remember, because it is recursive, you press
Ctrl-Break to stop the procedure.

Try **SPI** with other inputs.

Now let's change **SPI** and give it a third input
:INC (INCrease), which **SPI** will add to :STEP
instead of 2. Then you can change how much
the turtle's step increases by choosing different
numbers for the third input. Type

```
ED "SPI
```

The Logo editor shows this procedure.

```
TO SPI :STEP :ANGLE■
FD :STEP
RT :ANGLE
SPI :STEP + 2 :ANGLE
END
```

Now change the procedure so it looks like this.

```
TO SPI :STEP :ANGLE : INC
FD :STEP
RT :ANGLE
SPI :STEP + :INC :ANGLE :IN█
END
```

Now try
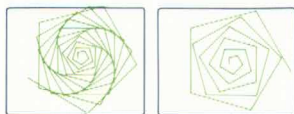
SPI 5 75 1
SPI 5 75 2

You may want to stop the turtle at different places. Try other inputs.

Chapter 14. The Turtle's Field

Contents

In Chapter 2, we learned that the turtle has a *position* and a *heading*. The turtle's heading can be described in degrees like a compass reading, with 0 or north at the top of the screen, 90 degrees directly east (right), 180 degrees directly south (bottom), and 270 degrees directly west (left). We could mark the screen:

```
                    0
                  North


    270                        90
    West                      East



                  South
                   180
```
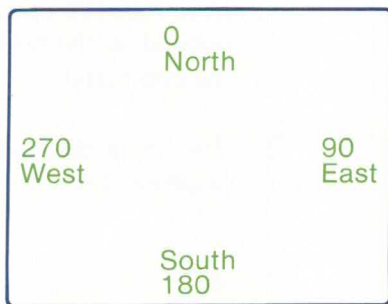
SETHEADING Command

When the turtle starts up, its heading is 0. After CLEARSCREEN, the turtle also has a heading of 0. Set the heading of the turtle in a particular direction by using the command SETHEADING (or SETH). Try

```
CS
SETHEADING 90
RIGHT 90
SETHEADING 90
```

▷	▽	▷

CS   RIGHT 90   SETH 90
SETH 90

Notice that **SETHEADING** acts differently
from **RIGHT** or **LEFT**. The end result does not
depend on the turtle's heading when you typed
the command.
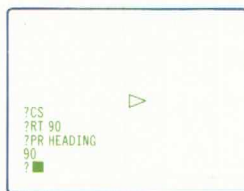
The turtle always knows which way it is
heading. Try

```
?CS           ▷
?RT 90
?PR HEADING
90
?■
```

CS
RT 90
PR HEADING

Logo responds

90

HEADING Operation

HEADING tells you the turtle's current
direction. It is a different kind of primitive from
PRINT or **FORWARD** because it does not
cause something to happen, but rather outputs
something which can be used as an input. The
color primitives **BACKGROUND**, **PALETTE**,
and **PENCOLOR** do this, too. These primitives
are called *operations* instead of commands.
Commands order Logo to do something.
Operations output something and are used as
inputs to commands or to other operations. In
this chapter, several other operations are
introduced.

POS Operation

POS tells you the turtle's position. This is described by two numbers that indicate how far the turtle is from the center. For example, the turtle's position at the start is [0 0].

Bug Box

To try out the operations described here, always give the command **PRINT** before the operation. If you don't you'll get an error message. For example, if you type

```
CS
PRINT POS
```

Logo responds

```
0 0
```

but if you type

```
POS
```

Logo responds

```
I DON'T KNOW WHAT TO DO WITH [0 0]
```

XCOR Operation

The first number that POS outputs indicates the turtle's location along an invisible horizontal line through the center of the screen or x-axis. This number is called the turtle's *x-coordinate*. If the turtle is west (left) of the center, then the number is negative.

YCOR Operation

The second number that POS outputs indicates the turtle's location along an invisible vertical line through the center of the screen or y-axis. This number is called the turtle's *y-coordinate*. If the turtle is south of (below) the center, then the number is negative.

The turtle, when at the center of the screen, has both an x-coordinate, XCOR, and a y-coordinate, YCOR, equal to 0. The screen dimensions are approximately as follows:
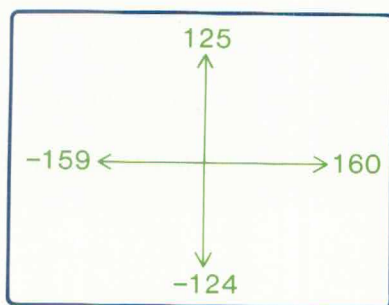
Find out the turtle's current position by asking Logo to PRINT POS.

If you type
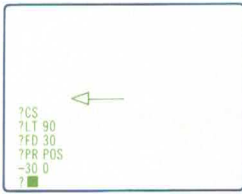
```
CS
LT 90
FD 30
PR POS
```

Logo responds

```
-30 0
```

The turtle is 30 steps west (left) of the center along the horizontal x-axis.

If you type
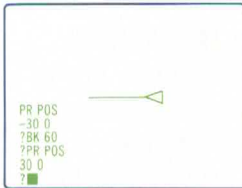
```
BK 60
PRINT POS
```

Logo responds

```
30 0
```

Now the turtle is 30 steps east (right) of the center along the horizontal x-axis.

You could also find out either coordinate by itself. Type

```
RT 90
FD 52
PR XCOR
```

Logo responds with the present x-coordinate.

30

Type

PR YCOR

Logo responds with the present y-coordinate.

52

The turtle is 30 steps east of the center and 52 steps north of the center.

SETPOS, SETX, and SETY Commands

POS, XCOR, and YCOR, like HEADING are all operations in Logo's vocabulary. There are also some commands that set the turtle at a specific position on the screen. They are SETPOS, SETX, and SETY. Notice that these commands do not act like FORWARD or BACK. The result does not depend on the position of the turtle when you typed the command. Also, these commands do not change the turtle's heading.

```
?FD 52
?PR XCOR
30
?PR YCOR
52
?■
```
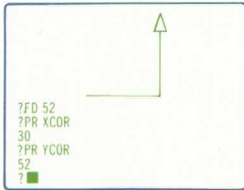
SETPOS Command

SETPOS sets the turtle at the position named by the input. The input must contain two numbers, the x-coordinate and the y-coordinate, and it must be enclosed [ ] (brackets). Type
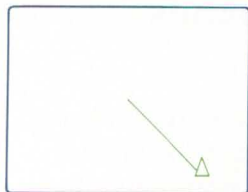
SETPOS [50 -52]

Be sure to leave a space before the -52.

This moves the turtle to 50 steps east of the center and 52 steps south. Notice that the turtle draws a line when moving to the point named by the input to SETPOS. To move the turtle without leaving a line, type **PENUP** before giving the **SETPOS** command.

SETX and SETY Commands

SETX and SETY each take one number as input: the x- or y-coordinate. They will set the turtle at that position along the specified x- or y-axis and leave the other coordinate as it was. For example, type
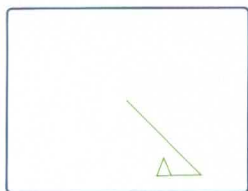
SETX 30

This moves the turtle horizontally to 30 steps east of the center, keeping the same y-coordinate.

PR POS
30 -52

The turtle is now on the southeast side: 30 steps east and 52 steps south of the center.

To move the turtle to a point specified by
SETX or SETY without leaving a line, type
PENUP before giving the SETX or SETY
command.

Other Turtle Commands

WRAP Command

When Logo starts up, the turtle starts out being
able to *wrap*. This means it can walk off one
edge of the screen and appear at the opposite
edge. It does not change direction. It's as if the
turtle just walked around the back of the screen
and wrapped an invisible line around it. You
may have seen an example of this earlier if you
used a very large input for a command such as
FORWARD or BACK. Type
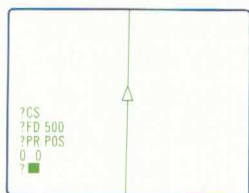
CS
FD 500
PR POS

Logo responds

0 0

Notice that the turtle is back at its home
position.

FENCE Command

The screen boundaries can also be set up so the turtle cannot move off the screen by typing

FENCE

Now type

CS
FD 500

Logo responds

TURTLE OUT OF BOUNDS
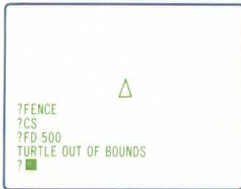
The turtle behaves as if there is a fence around the entire screen, and it cannot move out of these boundaries. It stays at its current position until given a command it can obey. The turtle screen will do this until you type

WRAP
CS
FD 500

Now the turtle is able to wrap around the screen as it did before.

```
?FENCE
?CS
?FD 500
TURTLE OUT OF BOUNDS
?■
```

```
?WRAP
?CS
?FD 500
?■
```
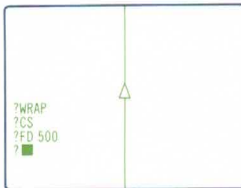
WINDOW Command

The WINDOW command allows the turtle to move off the screen without wrapping. So, although you can't see the turtle, it still carries out your orders along the x- and y-coordinates.

```
?CS
?WINDOW
?FD 500
?PR POS
0 500
?■
```

```
CS
WINDOW
FD 500
PR POS
```

Logo responds

0 500

The turtle is now 500 steps from the center and out of view. CS always restores the turtle to its center position.

Try using SPI after typing FENCE, after typing WINDOW, and then again after typing WRAP.

```
?CS
?FENCE
?SPI 5 125 2
TURTLE OUT OF BOUNDS IN SPI
FD :STEP
?■
```
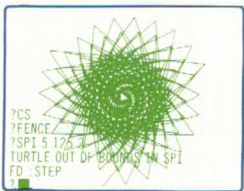
FENCE

```
CS
FENCE
SPI 5 125 2
```
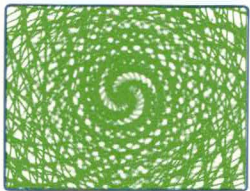
WINDOW

```
CS
WINDOW
SPI 5 125 2
```

Press Ctrl-Break when you want to stop.
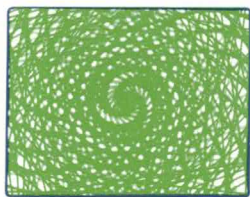
Using POS to Draw

There is an easy way to draw a *right triangle* (triangle with a 90-degree or right angle in it) if you know the length of the two sides joining in a right angle. First record the starting position of the turtle. Do this by using the Logo command MAKE. Type

```
CS
MAKE "START POS
```

MAKE Command

MAKE takes two inputs. The first one must be a name usually preceded by " (quotation mark). It is the name by which you tell Logo to remember the second input.

Here, MAKE has two inputs: START and POS. First, it creates a container or variable called START in your workspace. Second, it puts into this container the information passed from the operation POS. Once the container is created, you can look at its contents by putting a : (colon) in front of its name. Thus, if you typed

```
PR :START
```

(reading this as "print the contents of the container START"), and the turtle was in the center of the screen when you typed MAKE "START POS, Logo responds

0 0

Now have the turtle draw the two sides with the right angle in between. Type

```
FD 33
RT 90
FD 42
```

Now use the command SETPOS to bring the turtle back to its starting position, which is stored in the variable START. Type

```
SETPOS :START
```
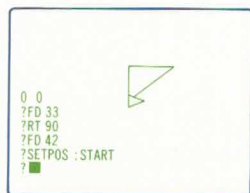
The turtle is moved to :START and because the pen is down, a line is drawn. Type the following procedure for this type of triangle:

```
0 0
?FD 33
?RT 90
?FD 42
?SETPOS :START
?■
```

```
TO TRI :SIDE1 :SIDE2
MAKE "START POS
FD :SIDE1
RT 90
FD :SIDE2
SETPOS :START
END
```

Try
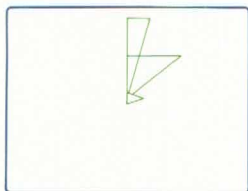
```
CS
TRI 75 20
SETH 0
TRI 40 50
```

Logo Vocabulary

Here is a list of commands and operations introduced in this chapter, along with their short forms.

Command	Short Form
FENCE	
MAKE	
SETHEADING	SETH
SETPOS	
SETX	
SETY	
WINDOW	
WRAP	

Operation
HEADING
POS
XCOR
YCOR

Chapter 15. Interacting with the Computer

Contents

You will write many types of interactive programs once you've become acquainted with the computer. For example, have Logo ask questions and receive answers in words or sentences. You may want to put Logo into action by a touch of a key. This requires using the operation READCHAR (or RC).

READCHAR (or RC) Operation

READCHAR reads the character typed on the keyboard and passes it to another primitive. In the following example, the character is passed to PRINT. Type

PRINT RC

RC makes Logo wait for a key to be pressed. Press the key A.

When you have pressed it, Logo does not wait for you to type anything else. It acts immediately, and **PRINT** displays the character typed.

A

RC is an operation like **HEADING** or **POS**. It is used as an input to another command or operation. For example, we could make Logo remember RC's output by using MAKE. Type

MAKE "KEY RC

Remember to put a " (quotation mark) in front of **KEY**.

You have told Logo to create a variable called KEY that will contain the output of RC.

Now type the character Z. The character does not appear on the screen when you type it. In other words, Logo does not "echo" on the screen what you type to it because you have not told Logo to print the character.

KEY will contain the character Z. To check this out, type

PRINT :KEY

Precede KEY with a : (colon) to tell Logo to print the contents of KEY.

Logo will respond

Z

Some Logo Grammar

If you type RC (followed by pressing the Enter (←) key) and then type a character like X. Logo responds

I DON'T KNOW WHAT TO DO WITH X

Primitives and procedures can be commands or operations. Commands order Logo to do something. Operations output something and are used as inputs to commands or to other operations. They cannot be used by themselves.

Turtle Drawing with the Touch of a Key

Let's define the procedure DRAW. We'll set it up so the following will happen:

F makes the turtle move forward 20 steps.
R makes the turtle turn right 30 degrees.
L makes the turtle turn left 30 degrees.

```
TO DRAW
MAKE "ANSWER RC
IF :ANSWER = "F [FD 20]
IF :ANSWER = "R [RT 30]
IF :ANSWER = "L [LT 30]
DRAW
END
```

DRAW waits until a key is pressed. When a key is pressed, the character is placed inside the variable ANSWER. DRAW then checks to see if ANSWER contains F. If so, then FD 20 is carried out. DRAW goes on to check if ANSWER contains R and the turtle will do a RT 30 if it does. Finally, it checks whether ANSWER contains L and does a LT 30 if it does.

IF Command or Operation

This procedure introduces a new primitive called IF. IF means the same thing in Logo as it does in English. It checks to see whether the statement that follows it is true. If so, then the instructions that follow in the [ ] (brackets) are carried out.

The procedure DRAW is recursive, like many other procedures we have learned. The last line of the procedure DRAW causes DRAW to be carried out again. The procedure does not stop until you press Ctrl-Break.

Try it. Make turtle designs all over the screen by using the F, L, and R keys. Type

```
DRAW
```

KEYP Operation

Let's make it more exciting by keeping the turtle in constant motion. In DRAW, the turtle sits and waits until you press a key. In the following procedures, the turtle keeps moving forward, changing direction only if you press a key. Type

```
TO DRIVE
IF KEYP [LISTEN]
FD 1
DRIVE
END

TO LISTEN
MAKE "ANSWER RC
IF :ANSWER = "R [RT 15]
IF :ANSWER = "L [LT 15]
END
```

DRIVE checks to see if a key has been pressed. The primitive KEYP outputs TRUE if a key has been pressed. IF then calls the procedure LISTEN. LISTEN places the character typed into the variable ANSWER. If ANSWER contains R or L, RT 15 or LT 15 is carried out before LISTEN ends. Then Logo returns to the second instruction in DRIVE. This instruction tells the turtle to FORWARD 1. The last instruction in DRIVE is recursive, causing DRIVE to repeat until you press Ctrl-Break.

If a key was not pressed, KEYP outputs FALSE, the turtle goes FD 1, and then DRIVE repeats as above. Use the DRIVE procedure to draw all over the screen.

Conditions and Actions

IF is a very useful primitive. It needs two inputs: a condition and an action that is carried out if the condition is TRUE. The action is a list of Logo instructions enclosed in [ ] (brackets) (like the second input to REPEAT). The condition may be given by a special kind of operation called a *predicate*. A predicate is a word that asks whether something is TRUE or FALSE. Each predicate in Logo ends with the letter P for predicate. In the DRIVE example just given, KEYP was a predicate asking whether a key had been pressed.

15-7

In the statement

IF KEYP [LISTEN]

the condition is the predicate

KEYP

and the action is the procedure

LISTEN

KEYP outputs TRUE if a character has been typed at the keyboard and FALSE if not. If a character has just been typed, the condition is TRUE and the action LISTEN (the procedure) is carried out. If KEYP outputs FALSE, the action is not carried out.

In the statement

IF :ANSWER = "F [FD 20]

the condition is

:ANSWER = "F

and the action is

[FD 20]

If ANSWER contains the character F, the condition is TRUE and the action FD 20 is carried out. If the condition is FALSE, the action is not carried out.

= (Equal) Operation

The = (equal) operation compares two inputs, one that precedes it and one that follows it. It outputs TRUE when the inputs are the same; it outputs FALSE when they are not.

Logo Vocabulary

Here is a list of commands and operations introduced in this chapter, along with their short forms.

Command	Short Form
IF	

Operation	Short Form
IF	
KEYP	
READCHAR	RC
=	

Chapter 16. A Game Project

Contents
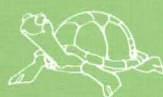
Let's create a game now. A target and a turtle appear somewhere on the screen. You are the player. Try to get the turtle into the target with the least number of moves.

For a first version, the moves will be Logo commands like LT 45 or FD 80. Later we will make the game even better by assigning keys to direct the turtle. As you develop the game and bring it to different stages, you will learn how to plan a project.

First, set up a target: then set up the turtle. One procedure will work for both tasks. An example is printed below. SETUP sets the turtle up in a random position on the screen, which means the turtle could begin anywhere. The turtle ends up heading 0 degrees, or north.

Type all of the following procedures as we describe them.

```
△

?SETUP
?■
```

SETUP

```
TO SETUP
PU
RT RANDOM 360
FD RANDOM 100
SETHEADING 0
PD
END
```

16-3

RANDOM Operation

The Logo operation RANDOM outputs a
number that Logo makes up, so you never know
what it is. This number is always less than the
number given to RANDOM as its input.

In SETUP, the turtle turns RIGHT some angle,
which can be as small as 0 degrees or as large
as 359 degrees. The actual number is computed
each time RANDOM is used. The input to FD
is also a random number. Here, the number can
be no larger than 99 because the input to
RANDOM is 100.

Notice that the next-to-last instruction in
SETUP leaves the turtle facing north.

SETUP can be used to set up the turtle as well
as the target. Put the turtle back in the center
after drawing the target and before setting up the
turtle's position.

The procedure SETGAME sets up the game.

```
TO SETGAME
CS
SETUP
TARGET
PU
SETPOS [0 0]
SETUP
END
```

SETGAME calls SETUP to set up the target's
position and then calls TARGET to draw the
target at that position.

```
TO TARGET
BOXR 10
END
```

SETGAME then calls SETUP to set the
turtle's first position. Use SETGAME a few
times and try each time to make the turtle hit the
target by using turtle commands. It's hard at
first.
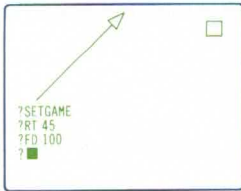
```
SETGAME
RT 45
FD 100
```

Did you hit the target?

Using a Key as a Game Button

The main purpose of this game is to move the
turtle around by the touch of a key as we did in
the last chapter. If we decide that the following
characters will cause the turtle to do certain
things,

F for Forward 10,
R for Right 15, and
L for Left 15,

then we can use READCHAR (or RC) to read
the character typed at the keyboard. We ask
Logo to remember which key was pressed by
giving it a name.

16-5

Now we can refer to this character by
:ANSWER. Here is the procedure. (Edit your
DRAW procedure to save typing.)

```
TO PLAY
MAKE "ANSWER RC
IF :ANSWER = "F [FD 10]
IF :ANSWER = "R [RT 15]
IF :ANSWER = "L [LT 15]
PLAY■
END
```
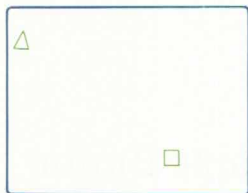
TO PLAY
MAKE "ANSWER RC
IF :ANSWER = "F [FD 10]
IF :ANSWER = "R [RT 15]
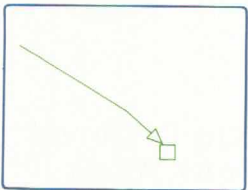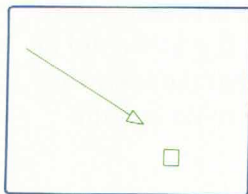IF :ANSWER = "L [LT 15]
PLAY■
END

Don't forget to set the target before trying the
procedure!

SETGAME
PLAY

Typing F, R, or L will make the turtle move.
Typing other characters will have no effect. You
must press Ctrl-Break to stop the procedure
PLAY.

Expanding the Game Project

In this section, we'll build a better target game out of SETGAME and PLAY. Some of the techniques used in this game may be new, while others are not. We can make a procedure GAME that uses SETGAME and then PLAY.

```
TO GAME
CS
SETGAME
PLAY
END
```

Try GAME.

```
GAME
```

Perhaps we should raise the turtle's pen before we play. It also would be nice if GAME printed some instructions. Edit the GAME procedure and type in the RULES procedure.

```
TO GAME
CS
RULES     (GAME calls the procedure RULES)
SETGAME
PU
PLAY
END
```

```
TO RULES
PRINT [HIT THE TARGET WITH THE TURTLE]
PRINT [TYPE R OR L TO TURN AND F TO ADV→
ANCE]
END
```

Notice that PRINT displays messages as well
as values of an operation. Remember, however,
to enclose the words you want printed in
[ ] (Brackets).

Try GAME now.

```
GAME
```

This is much better, but there is still room for
improvement! The game is too slow. Let's make
it more challenging.

Why not give the player only one chance to land
on the target? The player can turn the turtle
many times but will have only one chance to tell
it how far to go forward.

Here is the plan: after Logo sets up the game,
we want it to let you play the game. Once
you've made your try, you can see if you landed
on target. Logo leaves that picture on the screen
for a little while and then starts the game again
with a new target and position.

We'll use a *top-down* approach to plan this
game. That means we'll write the overall
structure of the game (the superprocedure)
before we know how to write all of the details
(what goes into the subprocedures).

```
TO GAME
CS
RULES
SETGAME    (This sets up each game.)
PU
PLAY
WAIT 100   (Logo waits a little while. Now start a
GAME        new game.)
END
```

Edit the procedure PLAY to give yourself only one chance to move the turtle forward into the target. The object of the game is to judge the distance. When you press the T key (T for try), you get your chance to land on the target. Your PLAY procedure should now look like this:

```
TO PLAY
MAKE "ANSWER RC
IF :ANSWER = "R [RT 15]
IF :ANSWER = "L [LT 15]
IF :ANSWER = "T [TRYLANDING STOP]
PLAY
END
```

STOP Command

The STOP command in the third IF statement is very important. It makes the procedure PLAY stop after it calls the procedure TRYLANDING. Otherwise, you would have to interrupt the whole game by pressing Ctrl-Break to stop this recursion.

Edit RULES and change F to T.

```
TO RULES
PR [HIT THE TARGET WITH THE TURTLE]
PR [TYPE R OR L TO TURN AND T TO TRYLAN→
DING]
END
```

We've used the *top-down* approach again. We've changed PLAY to use a procedure named TRYLANDING, which isn't defined yet! Let's define it now.

```
TO TRYLANDING
PR [HOW FAR DO YOU WANT TO MOVE FORWARD→
?]
FD READWORD
END
```

READWORD (or RW) Operation

READWORD (or RW) is like RC except you type a word instead of a single character. Logo waits for you to press the Enter ⏎ key to signal that you are done. READWORD outputs the word typed, and FORWARD moves the turtle forward that amount.

Now we have written the whole game. To try it, type

```
GAME
```

Remember to give the commands R and L to turn the turtle and T to try landing on the target when you are ready. After you type T, Logo waits for you to type a number and then press Enter ⏎ .

You may be able to adapt this game and the techniques used in it to make other games that interact with the computer. You can also add improvements to this game. For example, have Logo figure out whether you landed on the target. Logo could also keep track of your score.

Logo Vocabulary

Here is a list of commands and operations introduced in this chapter, along with their short forms.

Command	Short Form
STOP	

Operation	Short Form
RANDOM	
READWORD	RW

Chapter 17. Recursive Procedures

Contents

One of the most powerful features of Logo is that you can divide a project into procedures. Each of these procedures has its own name and is a separate piece. A procedure can call or be called (used) by any other procedure. Recursive procedures call themselves. For example, POLY and SPI are both recursive.

```
TO POLY :STEP :ANGLE
FD :STEP
RT :ANGLE
POLY :STEP :ANGLE                    (Recursive)
END
```

```
TO SPI :STEP :ANGLE :INC
FD :STEP
RT :ANGLE
SPI :STEP + :INC :ANGLE :INC (Recursive)
END
```

POLY calls POLY as part of its definition and SPI calls SPI.

Let's think about this process. Imagine that Logo has an unlimited supply of helpers living in the computer. Every time a procedure is called, a helper looks up the definition of the procedure. The helper begins to carry out the instructions by calling on other helpers. Usually, several helpers are needed to carry out one procedure.

For example, when POLY is called, its helper calls a FD helper. After the FD helper finishes, a RT helper is called. When it finishes, another POLY helper is called. The second POLY helper calls a FD helper, a RT helper, and a third POLY helper. Meanwhile, the first POLY helper is still waiting for the second POLY helper to finish. At some point, the FD helpers and the RT helpers finish their jobs (we don't know who they call for help). The POLY helpers never finish; they keep on calling for new POLY helpers.

When you use POLY, the process continues until you press Ctrl-Break. Not all recursive procedures work this way. They can be made to stop. In fact, making up appropriate *stop rules* is an important part of writing recursive procedures.

Stop Rules

There are many kinds of stop rules we could make up. Here is a simple example of a fast way to stop SPI.

```
TO SPI :STEP :ANGLE :INC
IF KEYP [STOP]
FD :STEP
RT :ANGLE
SPI :STEP + :INC :ANGLE :INC
END
```

KEYP is a predicate that outputs TRUE when any key on the keyboard is pressed. Using it with IF lets you stop SPI by pressing any key.

Here is a different kind of stop rule.

We could decide that SPI should stop if :STEP
is greater than 200. So, we replace the first line
with

```
IF :STEP > 200 [STOP]
```

The > means *greater than* to Logo.

After editing, SPI looks like this:

```
TO SPI :STEP :ANGLE :INC
IF :STEP > 200 [STOP]
FD :STEP
RT :ANGLE
SPI :STEP + :INC :ANGLE :INC
END
```

Try the new SPI. If you don't like the stop rule,
change it.

Making up a stop rule for POLY can be a little
trickier. POLY completes a figure when the
turtle returns to its starting state. This means
that the turtle must turn a complete rotation of
360 degrees or a multiple of 360.

In any case, we need to know what the turtle's heading was when it started and compare it to the turtle's heading after each turn to know when to stop. So before **POLY** is called, we have to know the turtle's heading. Type

```
MAKE "START HEADING
```

POLY can check to see if the turtle's current heading is the same as :START.

IF HEADING = :START [STOP]

When you put the stop rule into the procedure, make sure that it's placed after the **RT** command. What will happen if you put it before? The procedure will stop right away – even before the turtle starts drawing. Now type

```
EDIT "POLY
```

Insert the stop rule and press the (Esc) key to leave the editor.

```
TO POLY :STEP :ANGLE
FD :STEP
RT :ANGLE
IF HEADING = :START [STOP]
POLY :STEP :ANGLE
END
```
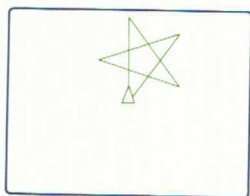
Now try **POLY**. Remember that it needs two inputs.

There is a problem here. If you forget to tell Logo to remember the starting heading before you run **POLY**, **POLY** does not work.

POLY 40 144

17-6

It is best to put that action into a procedure. Let's rename POLY and call it POLY1. Then we can make a new POLY, which sets up :START and then runs POLY1. For example, make the following changes:

```
TO POLY :STEP :ANGLE
MAKE "START HEADING
POLY1 :STEP :ANGLE
END

TO POLY1 :STEP :ANGLE
FD :STEP
RT :ANGLE
IF HEADING = :START [STOP]
POLY1 :STEP :ANGLE
END
```

Now POLY will do the whole job. HEADING is an operation so we put no : (colon) in front of it as we do to indicate the contents of a variable.

You can create all sorts of new and dazzling designs with SPI and POLY because there are so many ways to combine spirals and polygons.

A Different Kind of Recursion

One of the many interesting kinds of spirals you can play with makes only 90-degree turns. Let's call it SQUIRAL. Type

```
TO SQUIRAL :STEP :INC
IF :STEP > 80 [STOP]
FD :STEP
RT 90
SQUIRAL :STEP + :INC :INC
END
```
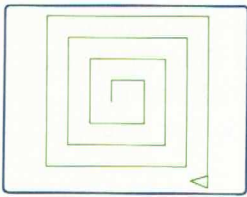
SQUIRAL 10 5

SQUIRAL 20 2

Now try

```
SQUIRAL 10 5
```
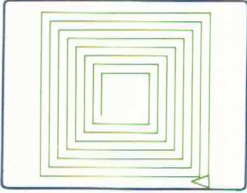
```
SQUIRAL 20 2
```

This is nice, but let's try something different! Add a few more turtle actions after the recursion line and see what happens. For example, add

```
BK :STEP
LT 90
```

at the end of the SQUIRAL procedure.

Let's edit SQUIRAL and change the title to SURPRISE.

```
TO SURPRISE :STEP :INC
IF :STEP > 80 [STOP]
FD :STEP
RT 90
SURPRISE :STEP + :INC :INC
BK :STEP
LT 90
END
```
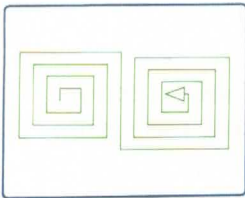
SURPRISE 10 5

SURPRISE 42 2

Now try it.

SURPRISE 10 5
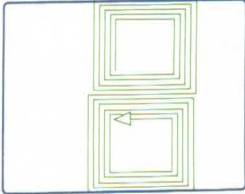
SURPRISE 42 2

If you are puzzled by the effect, imagine a whole supply of helpers lined up in the computer.

A recursive procedure calls a series of helpers to do the work. Each of these helpers is waiting for the one it called to finish. It then must go on and finish its own work. In the examples given of recursive procedures, there was no more work to do after the recursion. Here, however, each helper must cause the turtle to move BACK and then turn LEFT before it can quit and tell whoever it is helping to return to work.

Logo Vocabulary

Here is a list of operations introduced in this chapter.

Operation

> (greater than)

A Concluding Note

This introduction to Logo is finished, but we hope you will explore other turtle projects on your own.

The *Logo Reference* describes more advanced graphics features as well as many other features of Logo that you may want to try now.

Have fun!

Appendixes

Contents

A-1
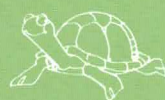
Appendix A. Before You Begin

When you use your Logo Language Diskette for the first time, there are certain procedures you must follow. They are:

1. Copying the Logo Language Diskette

2. Creating a file diskette.

These procedures have to be done only once when you start using Logo. They are not difficult if you follow the instructions step by step. After your first use of Logo, you can skip this section.

Copying the Logo Language Diskette

You should make a backup copy, store the master in a safe place, and always use the backup copy. If you damage or lose your backup copy, you can use your master to make a new backup copy.

To make a copy, you need a blank diskette. Make sure your Logo Language diskette has a write-protect tab covering the notch on the diskette. This is important, because if the diskettes get mixed up, this message is displayed:

```
Target diskette write protected
Correct, then strike any key
```

If you get this message, do the following:

1. Remove the Logo Language Diskette from the drive.

2. Insert the backup diskette into the drive.

3. Press any key. (You do *not* have to press the Enter (←) key.)

Before you begin copying your Logo Language Diskette, it is important to check whether you have one drive or two drives. (See the illustration below.)

> Note: If you have an IBM Personal Computer *XT*, follow steps for one drive.

Drive A

Drive A   Drive B
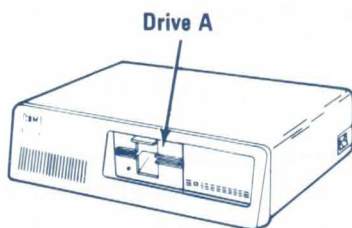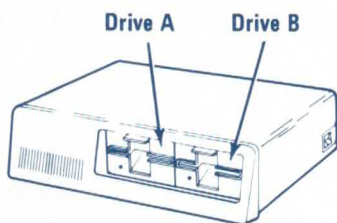
One Drive

Two Drives

One-Drive System

If you have one drive, do the following:

1. Start Logo as in steps 1 through 5 in the
   section "Starting Logo" in Chapter 1. If
   Logo is already started, make sure your
   Logo Language Diskette is in the drive.

2. Type .DOS and press the Enter ⟨←⟩ key.

3. You should see the following DOS prompt
   on your screen:

   A>

4. Type:

   DISKCOPY

   and press the Enter ⟨←⟩ key.

5. This message appears:

   Insert source diskette in drive A:

   Strike any key when ready

   Because the Logo Language Diskette (the
   source diskette) is already in drive A, you
   do not need to exchange diskettes.

6. Press any key. You will see this message:

Copying 9 sectors per track, 1 side(s)

7. You will see the "in use" light on your disk drive come on while the original diskette is being read, and then this message is displayed:

Insert target diskette in drive A:

Strike any key when ready

Before touching a key:

a. Remove the Logo Language Diskette.

b. Insert the backup diskette (the blank diskette). This is the *target diskette*.

c. Now press a key to tell **DOS** that the correct diskette has been inserted. You will see the message:

Formatting while copying

8. You will see the "in use" light on your disk drive come on while the backup diskette is being written. Then this message will be displayed:

Insert source diskette in drive A:

Strike any key when ready

Do this:

    a.   Remove the backup diskette.

    b.   Insert the Logo Language Diskette.

    c.   Press any key only after the Logo Language Diskette is in the drive.

9. Repeat steps 7 and 8 until this message appears:

Copy complete

Copy another (Y/N)?

10. Type:

N

You *don't* have to press the Enter ⟵ key.

The DOS prompt, A>, is displayed.

11. Remove the diskette from the drive. With a felt-tip pen, label and date your Logo Language Diskette Backup. Place a write-protect tab over the notch of the backup diskette; store the master in a safe place.

Skip the next section and go to "Creating a File Diskette."

Two-Drive System

If you have two drives, do the following:

1. Start up Logo as in steps 1 through 5 in the section "Starting Logo" in Chapter 1. If Logo is already started, make sure your Logo Language Diskette is in drive A.

2. Type .DOS and press the Enter ⬅ key.

3. You should see the following DOS prompt on your screen:

   A>

4. Insert the blank diskette (the backup) into drive B.

5. Type:

   DISKCOPY A: B:

   and press the Enter ⬅ key. This gives the command to copy the diskette in drive A onto the diskette that's in drive B.

6. You will see this message:

   Insert source diskette for drive A:

   Insert target diskette for drive B:

   Strike any key when ready

7. Press any key, because the backup diskette should already be in drive B. You will see this message:

Copying 9 sectors per track, 1 side(s)

Formatting while copying

Now the entire diskette is being copied from the diskette in drive A to the diskette in drive B. You will see one "in use" light go on and then the other.

8. When the copy has been made, you will see this message:

Copy complete

Copy another (Y/N)?

9. Type:

N

You *don't* have to press the Enter ⏎ key.

The DOS prompt, A>, is displayed.

10. Remove the backup diskette from drive B. With a felt-tip pen, label and date your Logo Language Diskette Backup. Place a write-protect tab over the notch of the backup diskette; store the master in a safe place.

Creating a File Diskette

In order to save the programs that you write, you will need a formatted diskette. Any blank diskette compatible with your IBM Personal Computer can be used as a Logo file diskette. Your Logo Language Diskette contains the IBM Personal Computer Disk Operating System (DOS) Version 2.00 and the FORMAT command. Only the FORMAT command from DOS 2.00 can be used to format a diskette compatible with the Logo language. Before beginning to format a diskette it is important to check whether you have one drive or two drives. (See the illustration below.)

Note: If you have an IBM Personal Computer *XT*, follow steps for one drive.

Drive A

One Drive

Drive A   Drive B
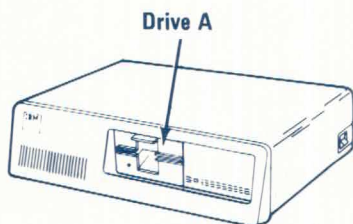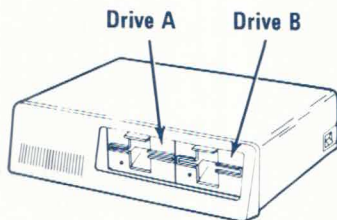
Two Drives

One-Drive System

If you have one drive, follow these instructions to format a diskette. If you already have the DOS prompt on your screen, skip Steps 1 and 2.

1. Start up Logo as in steps 1 through 5 in the section "Starting Logo" in Chapter 1. If Logo is already started, make sure your Logo Language Diskette is in drive A.

2. Type .DOS and press the Enter ⏎ key.

3. You should see the following DOS prompt on your scrren:

    A>

4. Type FORMAT A: and press the Enter ⏎ key.

5. You will see the following message:

    Insert new diskette for drive A:
    and strike any key when ready

6. When the drive "in use" light is off, remove your Logo Language Diskette. Holding your blank diskette so that the notch is on the left side near the front, insert it into the drive and close the door.

7. When your blank diskette is inserted into the drive and the door is closed, simply press any key. You will see the light on the drive come on, and the message

    Formatting...

    on the screen. In a few moments you will see the message:

Formatting...Format complete

xxxxxx bytes total disk space
xxxxxx bytes available on disk

Format another (Y/N)?

If you have more blank diskettes that you would like to format, then type Y and repeat steps 6 and 7 again.

8.   When you are finished formatting diskettes, type N.

9.   You should see the DOS prompt A> again on your screen. Remove the file diskette from the drive and put it aside for the moment. With a felt-tip pen, label and date your file diskette. You can now restart Logo by inserting your Logo Language Backup diskette into drive A, typing LOGO, and pressing the Enter ⟨←⟩ key.

A> LOGO

Two-Drive System

If you have two drives, follow these instructions to format a diskette. If you already have the DOS prompt on your screen, skip Steps 1 and 2:

1.   Start up Logo as in steps 1 through 5 in the section "Starting Logo" in Chapter 1. If Logo is already started, make sure your Logo Language Diskette is in drive A.

2.   Type .DOS and press the Enter ⟨←⟩ key.

3. You should see the following DOS prompt on your screen:

   A>

4. Type FORMAT B: and press the Enter ⬅ key.

5. You will see the following message:

   Insert new diskette for drive B:
   and strike any key when ready

6. Holding your blank diskette so the notch is on the left side near the front, insert it into drive B and close the drive door.

7. When your blank diskette is inserted into drive B and the door is closed, press any key. You will then see the light on drive B come on, and the message

   Formatting...

   will appear on the screen. In a few moments you will see the message:

   Formatting...Format complete

        xxxxxx bytes total disk space
        xxxxxx bytes available on disk

   Format another (Y/N)?

   If you have more blank diskettes that you would like to format, type Y and repeat steps 6 and 7 again.

8. When you are finished formatting diskettes, type N.

9. You should see the DOS prompt A> again on your screen. Remove the file diskette from drive B and put it aside for the moment. With a felt-tip pen, label and date your file diskette. You can now restart Logo by inserting your Logo Language Backup diskette into drive A, typing LOGO, and pressing the Enter (←) key.

A> LOGO

Index

R

S

IBM   The Personal Computer

Reader's Comment Form

Logo: Programming with
Turtle Graphics   1502229

Your comments assist us in improving the usefulness of
our publication; they are an important part of the input
used for revisions.

IBM may use and distribute any of the information you
supply in any way it believes appropriate without
incurring any obligation whatever. You may, of course,
continue to use the information you supply.

Please do not use this form for technical questions
regarding the IBM Personal Computer or programs for
the IBM Personal Computer, or for requests for
additional publications; this only delays the response.
Instead, direct your inquiries or request to your
authorized IBM Personal Computer dealer.

Comments:

Fold here

SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

IBM does not warrant that the functions contained in the program will meet your requirements or that the operation of the program will be uninterrupted or error free.

However, IBM warrants the diskette(s) or cassette(s) on which the program is furnished, to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of delivery to you as evidenced by a copy of your receipt.

LIMITATIONS OF REMEDIES

IBM's entire liability and your exclusive remedy shall be:

1. the replacement of any diskette(s) or cassette(s) not meeting IBM's "Limited Warranty" and which is returned to IBM or an authorized IBM PERSONAL COMPUTER dealer with a copy of your receipt, or

2. if IBM or the dealer is unable to deliver a replacement diskette(s) or cassette(s) which is free of defects in materials or workmanship, you may terminate this Agreement by returning the program and your money will be refunded.

IN NO EVENT WILL IBM BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE SUCH PROGRAM EVEN IF IBM OR AN AUTHORIZED IBM PERSONAL COMPUTER DEALER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

GENERAL

You may not sublicense, assign or transfer the license or the program except as expressly provided in this Agreement. Any attempt otherwise to sublicense, assign or transfer any of the rights, duties or obligations hereunder is void.

This Agreement will be governed by the laws of the State of Florida.

Should you have any questions concerning this Agreement, you may contact IBM by writing to IBM Personal Computer, Sales and Service, P.O. Box 1328-W, Boca Raton, Florida 33432.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU FURTHER AGREE THAT IT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN US WHICH SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN US RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.